Reconstructing Surfaces of Particle-Based Fluids Using Anisotropic Kernels

JIHUN YU
Industrial Light and Magic
and
GREG TURK
Georgia Institute of Technology

In this article we present a novel surface reconstruction method for particle-based fluid simulators such as smoothed particle hydrodynamics. In particle-based simulations, fluid surfaces are usually defined as a level set of an implicit function. We formulate the implicit function as a sum of anisotropic smoothing kernels, and the direction of anisotropy at a particle is determined by performing Principal Component Analysis (PCA) over the neighboring particles. In addition, we perform a smoothing step that repositions the centers of these smoothing kernels. Since these anisotropic smoothing kernels capture the local particle distributions more accurately, our method has advantages over existing methods in representing smooth surfaces, thin streams, and sharp features of fluids. Our method is fast, easy to implement, and our results demonstrate a significant improvement in the quality of reconstructed surfaces as compared to existing methods.

Categories and Subject Descriptors: I.3.7 [**Computer Graphics**]: Three-Dimensional Graphics and Realism—*Animation*; I.3.5 [**Computer Graphics**]: Computational Geometry and Object Modeling—*Physically based modeling*

General Terms: Algorithms

Additional Key Words and Phrases: Computer graphics, fluid simulation, surface reconstruction, physically based animation, anisotropic basis functions

This work is supported in part by NSF grants CCF-0917093, CCF-0728977, and CCF-0811485.
Authors' addresses: J. Yu (corresponding author), Industrial Light and Magic; email: jihun.yu@gmail.com; G. Turk, Georgia Institute of Technology.

DOI 10.1145/2421636.2421641
http://doi.acm.org/10.1145/2421636.2421641

ACM Reference Format:
Yu, J. and Turk, G. 2013. Reconstructing surfaces of particle-based fluids using anisotropic kernels. ACM Trans. Graph. 32, 1, Article 5 (January 2013), 12 pages.
DOI = 10.1145/2421636.2421641
http://doi.acm.org/10.1145/2421636.2421641

1. INTRODUCTION

It is becoming increasingly popular to create animated liquids using physics-based simulation methods for feature film effects and interactive applications. There exist two broad categories for simulation methods based on their different approaches to spatial discretization: mesh-based methods and mesh-free methods. In mesh-based methods, the simulation domain is discretized into mesh grids and the values of physical properties on grid points are determined by solving the governing equations. In mesh-free methods, on the other hand, the fluid volume is discretized into sampled particles that carry physical properties and that are advected in space by the governing equations. In recent years, mesh-free methods have become a competitive alternative to mesh-based methods due to various advantages such as their inherent mass conservation, the flexibility of simulation in unbounded domains, and ease of implementation. Among various mesh-free methods, Smoothed Particle Hydrodynamics (SPH) is the most popular approach for simulating fluid since it is computationally simple and efficient compared to others. In computer graphics, SPH has been successfully used for the simulation of free-surface fluids [Müller et al. 2003], fluid interface [Müller et al. 2005; Solenthaler and Pajarola 2008], fluid-solid coupling [Müller et al. 2004b, Lenaerts et al. 2008; Becker et al. 2009b], deformable body [Becker et al. 2009a], multiphase fluid [Müller et al. 2004a; Keiser et al. 2005; Solenthaler et al. 2007], and fluid control [Thürey et al. 2006].

Although SPH has been used to simulate various fluid phenomena, extracting high-quality fluid surfaces from the particle locations is not straightforward. Classical surface reconstruction methods have difficulties in producing smooth surfaces due to irregularly placed particles. Few researchers have successfully addressed this issue of reconstructing smooth fluid surfaces from particles. In this article, which is an expanded version of our earlier paper [Yu and Turk 2010], we propose a novel surface extraction method that significantly improves the quality of the reconstructed surfaces. Our new method can create smooth surfaces and thin streams along with sharp features such as edges and corners. The key to our method is to use a stretched, anisotropic smoothing kernel to represent each particle in the simulation. The orientation and scale of the anisotropy is determined by capturing each particle's neighborhood spatial distribution. We obtain the neighborhood distribution in the

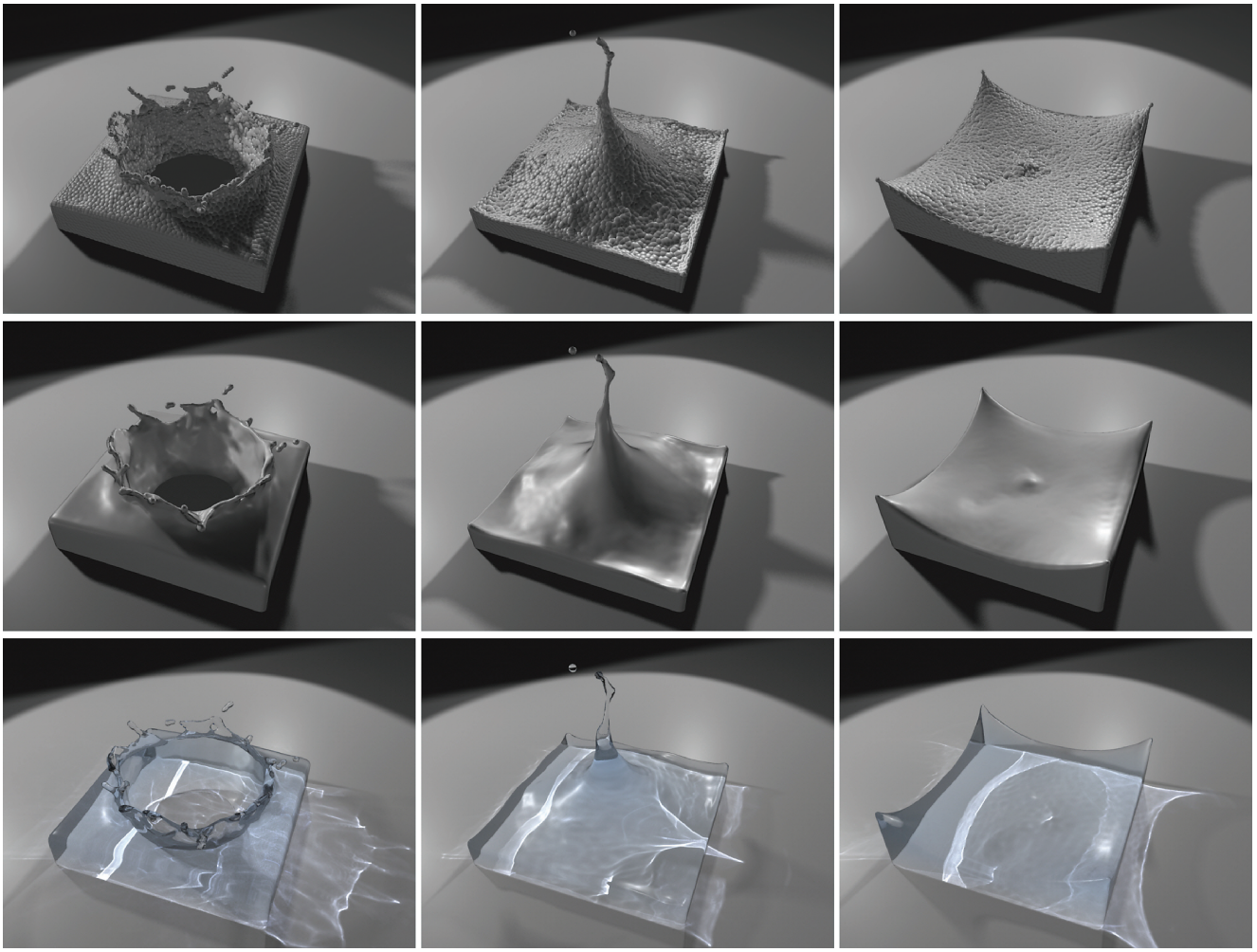


Fig. 1. Water splash. Top: Particle view. Middle: Opaque rendereing from our surface reconstruction. Bottom: Transparent rendering.

form of covariance tensor and analyze it through Principle Component Analysis (PCA). We then use these principal components to orient and scale the anisotropic kernel. We adjust the centers of these kernels using a variant of Laplacian smoothing to counteract the irregular placement of particles. A new density field is then constructed by the weighted mass contribution from the smoothing kernels. Finally, the renderable surface is reconstructed from the iso-surface of the given density field. We show that our new method leads to the realistic visualization of fluid surfaces and that it outperforms existing methods for handling smooth and thin surfaces with sharp features. The simplicity and efficiency of our method facilitates the incorporation of our method with existing SPH simulation schemes with little additional effort.

2. RELATED WORK

Because the surface representation of a fluid is crucial for realistic animation, methods for reconstructing and tracking fluid surfaces have been a topic of research since fluid simulation was first introduced in computer graphics. In mesh-based simulation frameworks, numerous methods have been proposed such as level-set methods [Osher and Fedkiw 2002], particle level-set methods [Enright et al. 2002a, 2002b, 2005], semi-Lagrangian contouring [Strain 2001; Bargteil et al. 2006], Volume-Of-Fluid methods (VOF) [Hirt and Nichols 1981] and explicit surface tracking [Brochu and Bridson 2006, 2009; Wojtan et al. 2009; Müller 2009].

In mesh-free (Lagrangian) simulation frameworks, Blinn [1982] introduced the classic blobby spheres approach. In this method, an iso-surface is extracted from a scalar field that is constructed from a sum of radial basis functions that are placed at each particle center. One of the drawbacks of Blinn's original formulation is that high or low densities of particles will cause bumps or indentations on the surface. Noting this problem, Zhu and Bridson [2005] modify this basic algorithm to compensate for local particle density variations. They calculate a scalar field from the particle positions that is much like a radial basis function that is centered at a particle. For a given location in space, they calculate a scalar value from a basis function whose center is a weighted sum of nearby particle centers, and whose radius is a weighted sum of particle radii. They then sample this scalar distance function on a grid, perform a smoothing pass over the grid, and then extract an iso-surface mesh from the grid. Their results are considerably smoother than the classic blobby spheres surface. Adams et al. [2007] further improved upon the method of Zhu and Bridson by tracking the particle-to-surface

distances over time. Specifically, they retain a sampled version of a signed distance field at each time step, and they use this to adjust per-particle distances to the surface. They perform particle redistancing by propagating the distance information from surface particles to interior particles using a fast marching scheme. The final surface is from the Zhu and Bridson scalar field, but calculated using these new per-particle distances. This method is successful at generating smooth surfaces both for fixed-radius and adaptively sized particles.

One drawback of these aforementioned methods is a typical assumption that the smoothing kernel of each particle is isotropic, and the spherical shape of the kernel makes it difficult to produce flat surfaces and sharp features. In contrast to these methods, our approach uses anisotropic kernels to stretch spheres into ellipsoids in order to alleviate those limitations.

Desbrun and Cani-Gascuel [1998] and Premoze et al. [2003] use a different surface tracking method in which a scalar field is advected on a Eulerian grid. Unfortunately this approach is more difficult to use for unbounded simulations.

Recently, an alternative method of surface reconstruction was proposed by Williams [2008]. In his method, a nonlinear optimization problem is solved iteratively to achieve global smoothness on surface mesh. An important contribution of the method is that the perfectly flat surfaces can be generated under certain conditions. Sin et al. [2009] use a level-set variant of the original method to alleviate the problem of temporal coherence. This method is extended in the work of Bhattacharya et al. [2011], in which they minimize thin-plate energy on an implicit level-set surface instead of smoothing an explicit mesh.

The idea of using anisotropic particles for surfacing of particle-based fluids was first reported in the work of Museth et al. [2007]. In their work, a similar technique to our approach was introduced where implicit ellipsoids are derived as eigenvalues and eigenvectors of a covariant matrix, and these ellipsoids are used to reduce the blobby look of the surfaces.

After the earlier version of our article was published [Yu and Turk 2010], Müller and Chentanez [2011] used the anisotropy concept presented in our original paper to create tight collision volumes for point-based solids, and this shows that this approach is applicable to solids as well. Our approach was also discussed in the work of Chris et al. [2011] as prior work on Lagrangian surface tracking.

Our anisotropic kernel approach is inspired by the work of Owen et al. [1995] and Liu et al. [2006]. They adapt anisotropic kernels to simulate large deformations of materials in the SPH framework, and their primary interest is in simulation accuracy. In their approach, the axes of their anisotropic kernels evolve in time according to the strain-rate tensor estimates. Our approach is also related to the work of Kalaiah and Varshney [2003] and Dinh et al. [2001]. Kalaiah and Varshney apply PCA to point clouds for point-based modeling. Dinh et al. reconstruct surfaces from voxel carving data by combining anisotropic kernels with variational implicit surfaces.

3. SPH FRAMEWORK

In SPH, the fluid volume is described as a set of particles with prescribed masses. In a given simulation step, physical quantities such as density and pressure are represented by values that are associated with each particle. For particle i at location $\mathbf{x}_i$, the density ρ_i is interpolated by a sum of the weighted contributions of nearby particle masses m_j.

$$\rho_i = \sum_j m_j W(\mathbf{x}_j - \mathbf{x}_i, h_j), \tag{1}$$

where W is the smoothing kernel and h_j is the smoothing radius associated with particle j. The pressure p_i of particle i is typically described as a function of the density of the fluid such as given by the Tait equation [Monaghan 1994], which is

$$p_i = k\rho_0 \left(\left(\frac{\rho_i}{\rho_0} \right)^{\gamma} - 1 \right), \tag{2}$$

where k and γ are stiffness parameters and ρ_0 is the rest density of the fluid. In the SPH framework, the Navier Stokes equation, discretized on particle locations, becomes an Ordinary Differential Equation (ODE) of the form

$$\rho_i \frac{\partial \mathbf{v}_i}{\partial t} = -\langle \nabla p \rangle(\mathbf{x}_i) + \mu \langle \Delta \mathbf{v} \rangle(\mathbf{x}_i) + \mathbf{f}_i^{ext}, \tag{3}$$

where $\mathbf{v}_i$ is the velocity, $\mathbf{f}_i^{ext}$ is an external force (such as gravity), μ is the viscosity constant, and $\langle \nabla p \rangle(\mathbf{x}_i)$ and $\langle \Delta \mathbf{v} \rangle(\mathbf{x}_i)$ are approximations of the pressure gradient and the velocity Laplacian at $\mathbf{x}_i$ in the SPH framework. For various approximations of differential operators in SPH framework, we refer the readers to Adams and Wicke [2009].

In this article, we simulate all of our examples using the Weakly Compressible SPH (WCSPH) framework [Becker and Teschner 2007]. Note that our method can be used with any SPH framework, as long as the simulator provides particle positions, radii, masses, and densities. In the case that mass and radius are global constants, our approach can be generalized to work with other Lagrangian simulation frameworks such as Particle-in-Cell (PIC) and Fluid-Implicit Particle (FLIP) [Zhu and Bridson 2005].

4. SURFACE RECONSTRUCTION

4.1 Surface Definition

Our surface definition is based on the approach proposed in Müller et al. [2003], where the surface is defined as an iso-surface of a scalar field

$$\phi(\mathbf{x}) = \sum_j \frac{m_j}{\rho_j} W(\mathbf{x} - \mathbf{x}_j, h_j), \tag{4}$$

and W is an isotropic smoothing kernel of the form

$$W(\mathbf{r}, h) = \frac{\sigma}{h^d} P\left(\frac{\|\mathbf{r}\|}{h} \right). \tag{5}$$

In the preceding equation, σ is a scaling factor, d is the dimension of the simulation, $\mathbf{r}$ is a radial vector, and P is a symmetric decaying spline with finite support. The scalar field $\phi(\mathbf{x})$ is designed as a normalized density field that smooths out the scalar value of 1 at each particle's position over a continuous domain, and an iso-surface from $\phi(\mathbf{x})$ gives a surface representation that coats the particles. However, the resulting surfaces often have bumps, and there are two reasons for this. First, the irregular placement of particles makes it difficult to represent an absolutely flat surface. Although irregular sampling is an essential feature of any Lagrangian scheme, this irregularity of the positions of the boundary particles can make surfaces appear blobby. Second, the spherical shape of the smoothing kernels is not suitable to describe the density distribution near a surface. That is, in order to correctly model surface geometry, it is necessary for the density of the near-surface particles to decrease at different rates in different directions.

To resolve the problem of irregular particle placement, we apply one step of diffusion smoothing to the location of the kernel centers. This process can be interpreted as a 3D variant of Laplacian

smoothing as described in Taubin [2000], and has the effect of denoising point clouds. The updated kernel centers $\bar{\mathbf{x}}_i$ are calculated by

$$\bar{\mathbf{x}}_i = (1-\lambda)\mathbf{x}_i + \lambda \sum_j w_{ij}\mathbf{x}_j / \sum_j w_{ij}, \tag{6}$$

where w is a suitable finite support weighting function and λ is a constant with $0 < \lambda < 1$. We use λ between 0.9 and 1 in our examples to maximize the smoothing effect. Note that this smoothing process is used only for surface reconstruction, and the averaged positions are not carried back into the simulation. Typically, Laplacian smoothing results in volume shrinking, and our approach also shrinks the fluid volume slightly by moving the kernels for boundary particles towards the inside. However, in contrast to level-set methods, our approach does not shrink volume continuously as the simulation evolves. Furthermore, the analysis in Appendix A shows that the maximum distance from our reconstructed surfaces to the original particle positions is within a small constant of the particle radius scale.

To cope with the problem of density distributions near the surface, our new approach is designed to capture the density distribution more accurately by allowing the smoothing kernels to be anisotropic. By replacing h with a $d \times d$ real positive definite matrix $\mathbf{G}$, we can simply redefine W to be an anisotropic kernel

$$W(\mathbf{r}, \mathbf{G}) = \sigma det(\mathbf{G}) P(\|\mathbf{G}\mathbf{r}\|). \tag{7}$$

The linear transformation $\mathbf{G}$ rotates and stretches the radial vector $\mathbf{r}$. Therefore $W(\mathbf{r}, \mathbf{G})$ becomes an anisotropic kernel, and iso-surfaces of W are ellipsoids instead of spheres. Note that the isotropic kernel can be treated as a special case of the anisotropic kernel by letting $\mathbf{G} = h^{-1}\mathbf{I}$ where $\mathbf{I}$ is an identity matrix. The key idea of our new method is to associate an anisotropy matrix $\mathbf{G}$ with each particle so that for particle j, $\mathbf{G}_j$ describes better the neighborhood density distribution.

Once all $\mathbf{G}_j$'s and $\bar{\mathbf{x}}_j$'s have been computed, we extract an iso-surface from a redefined scalar field

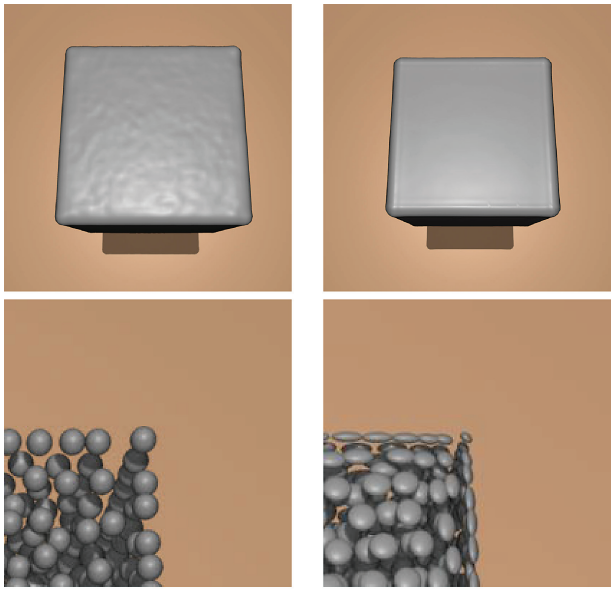
$$\phi_{new}(\mathbf{x}) = \sum_j \frac{m_j}{\rho_j} W(\mathbf{x} - \bar{\mathbf{x}}_j, \mathbf{G}_j). \tag{8}$$

It is necessary to point out that the equation of W is depending on the SPH simulation, since different SPH schemes can use different W's for density computation. For our examples, we use the B-cubic spline kernel from Becker and Teschner [2007].

4.2 Determining the Anisotropy

As mentioned in the previous subsection, our new approach determines an anisotropy matrix $\mathbf{G}$ for each particle in order to more accurately describe the density distribution around the particle. For example, in the neighborhood of a particle that is inside the fluid volume, the density is likely to be constant in all directions, making the corresponding $\mathbf{G}$ a scalar multiple of an identity matrix to keep the smoothing kernel W isotropic. On the other hand, around a particle that is near a flat surface, the particle density will decay faster along the normal axis than along the tangential axes. Then $\mathbf{G}$ should stretch W along the tangential axes and shrink W along the normal axis. At a sharp feature, the density will decay sharply in several directions, and $\mathbf{G}$ should shrink W in order to capture the sharp feature. See Figure 2 for a comparison between isotropic and anisotropic kernels that are near the surface of a region of fluid.

In order to determine $\mathbf{G}$, we apply the weighted version of Principal Component Analysis (WPCA) that is proposed in Koren and Carmel [2003] to the neighborhood particle positions. A drawback



(a) (b)

Fig. 2. Comparison between the surface reconstruction using isotropic kernels (a) and our anisotropic kernels (b). Top row: the surface of SPH particles from a single dam break simulation. Bottom row: Illustration of particles at the top right corner. The shape of a particle in (b) represents the anisotropy of the corresponding smoothing kernel. Note that our approach constructs a flat surface with sharp edges and corners from properly stretched particles.

of the conventional PCA is its sensitivity to outliers, and it often produces inaccurate information when the number of samples is small and the sample positions are noisy, which commonly happens in particle-based fluids. In contrast, WPCA achieves significant robustness towards outliers and noisy data by assigning appropriate weights to the data points. Specifically, WPCA begins by computing a weighted mean of the data points. Next, WPCA constructs a weighted covariance matrix $\mathbf{C}$ with a zero empirical mean and performs an eigendecomposition on $\mathbf{C}$. The resulting eigenvectors give the principal axes, and the eigenvalues indicate the variance of points along the corresponding eigenvalues. We then construct an anisotropy matrix $\mathbf{G}$ to match the smoothing kernel W with the output of WPCA.

In our approach, the weighted mean $\mathbf{x}_i^w$ and the covariance matrix $\mathbf{C}_i$ of particle i are formulated as

$$\mathbf{C}_i = \sum_j w_{ij}(\mathbf{x}_j - \mathbf{x}_i^w)(\mathbf{x}_j - \mathbf{x}_i^w)^T / \sum_j w_{ij}, \tag{9}$$

$$\mathbf{x}_i^w = \sum_j w_{ij}\mathbf{x}_j / \sum_j w_{ij}. \tag{10}$$

The function w_{ij} is an isotropic weighting function with respect to particle i and j with support r_i.

$$w_{ij} = \begin{cases} 1 - (\|\mathbf{x}_i - \mathbf{x}_j\|)/r_i)^3 & \text{if } \|\mathbf{x}_i - \mathbf{x}_j\| < r_i \\ 0 & \text{otherwise} \end{cases} \tag{11}$$

With the finite support of w_{ij}, the computation is confined to the neighborhood particles within the radius r_i. In our examples, we choose r_i to be $2h_i$ in order to include enough neighborhood

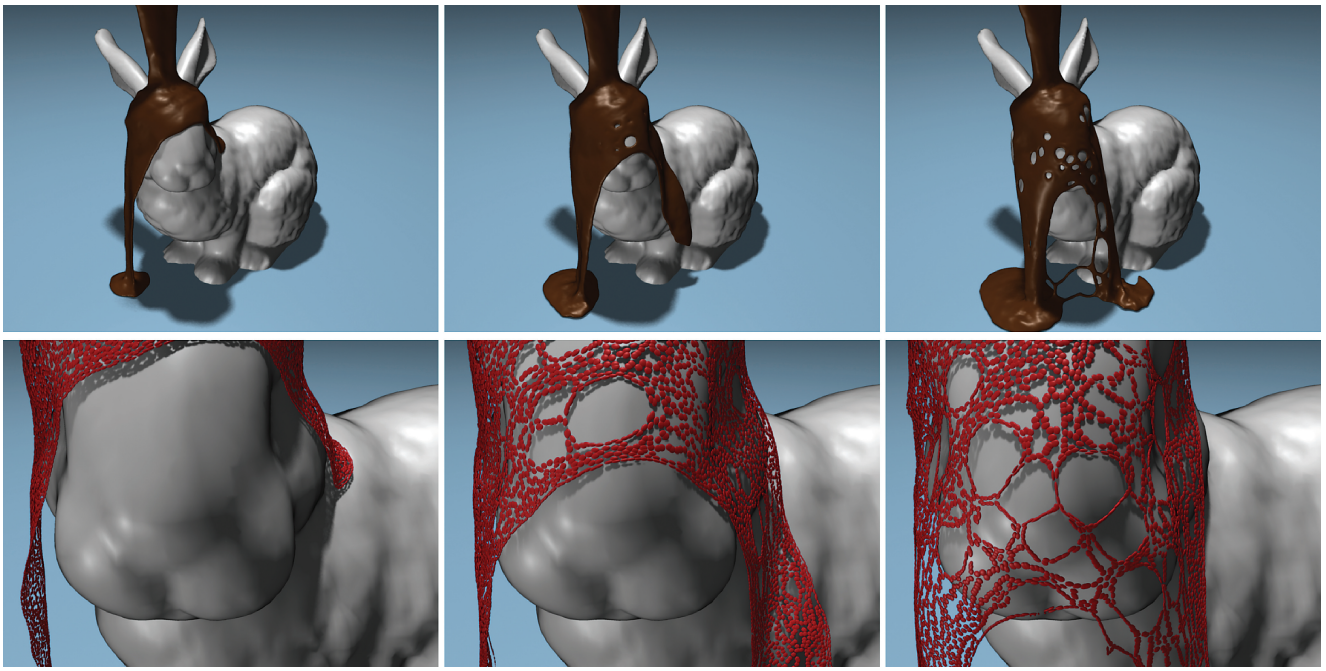


Fig. 3. Melted chocolate falling on a bunny.

particles and obtain reasonable anisotropy information. This kernel is also used to compute the averaged position of the particles in Eq. (6).

With each particle, the Singular Value Decomposition (SVD) of the associated $\mathbf{C}$ gives the directions of stretch or compression for deforming the smoothing kernel W in terms of eigenvectors and eigenvalues. The SVD yields

$$\mathbf{C} = \mathbf{R}\Sigma\mathbf{R}^T, \tag{12}$$

$$\Sigma = \mathrm{diag}(\sigma_1, \ldots, \sigma_d), \tag{13}$$

where $\mathbf{R}$ is a rotation matrix with principal axes as column vectors, and Σ is a diagonal matrix with eigenvalues $\sigma_1 \geq \ldots \geq \sigma_d$. In order to deal with singular matrices and prevent extreme deformations, we check whether $\sigma_1 \geq k_r\sigma_d$ with a suitable positive constant $k_r > 1$. This condition is true when the largest variance in one principal axis is much bigger than the smallest variance in another axis. In this case, we modify $\mathbf{C}$ so that the ratio between any two eigenvalues is with in k_r. Also, when the number of particles in the neighborhood is small, we reset W to a spherical shape by setting $\mathbf{G} = k_n\mathbf{I}$ in order to prevent poor particle deformations for nearly isolated particles. In addition, we multiply $\mathbf{C}$ by scaling factor k_s such that $\|k_s\mathbf{C}\| \approx 1$ for the associated particle inside fluid volume, in order to keep the volume of W constant for particles with the full neighborhood. The aforementioned processes are formulated as follows to obtain a modified covariance matrix $\widetilde{\mathbf{C}}$. We have

$$\widetilde{\mathbf{C}} = \mathbf{R}\widetilde{\Sigma}\mathbf{R}^T \tag{14}$$

$$\widetilde{\Sigma} = \begin{cases} k_s\mathrm{diag}(\sigma_1, \widetilde{\sigma}_2, \ldots, \widetilde{\sigma}_d) & \text{if } N > N_\epsilon, \\ k_n\mathbf{I} & \text{otherwise} \end{cases} \tag{15}$$

where $\widetilde{\sigma}_k = \max(\sigma_k, \sigma_1/k_r)$, N is the number of neighboring particles, and N_ϵ is a threshold constant. In our examples, we use $k_r = 4$, $k_s = 1400$, $k_n = 0.5$, and $N_\epsilon = 25$.

In order to make the kernel W of particle i deform according to $\widetilde{\mathbf{C}}_i$, $\mathbf{G}_i$ must be an inversion of $\widetilde{\mathbf{C}}_i$ and scaled by $1/h_i$ to reflect the original radius of particle i. Then our approach produces $\mathbf{G}_i$ as a symmetric matrix of the form

$$\mathbf{G}_i = \frac{1}{h_i}\mathbf{R}\widetilde{\Sigma}^{-1}\mathbf{R}^T. \tag{16}$$

4.3 Alleviating Attraction Artifact

Our particle relocation approach improves the qualify of fluid surfaces by denoising particle samples. However, as a side-effect, the particle relocation may introduce unphysical attraction effects between fluid components. These artifacts are mainly noticeable in the case where two separate fluid components are approaching one another. When two separate fluid components become closer than the radial support r_i of Eq. (11), particles in one component begin to classify particles in the other component as neighbors, and thus are moved closer to the other component by the relocation step. This artifact occurs because we use a large support $r_i = 2h_i$, where h_i is approximately twice the average particle spacing r_a. Therefore, the relocation step pulls particles together even when they are further apart than r_a, as long as they are within the range of $4r_a$.

In order to alleviate this problem, we use a Connected Component (CC) algorithm to mark simulation particles. We define two particles i and j as being connected if $\|\mathbf{x}_i - \mathbf{x}_j\| \leq r_a$. For a given particle, other connected particles are those neighboring particles within r_a. These nearby particles are easily identified by a neighborhood search. Starting with a seed particle, we compute its CC by a depth-first search graph traversal, and we label these connected particles with the index of the seed particle. To find all the CC's, we iterate through all the particles, and start a new traversal whenever we find a particle that has not yet been labeled. Once all of the particles have been labeled, we use a modified version of the weight w_{ij} of Eq. (11) that uses the CC information. Let the label of a particle i

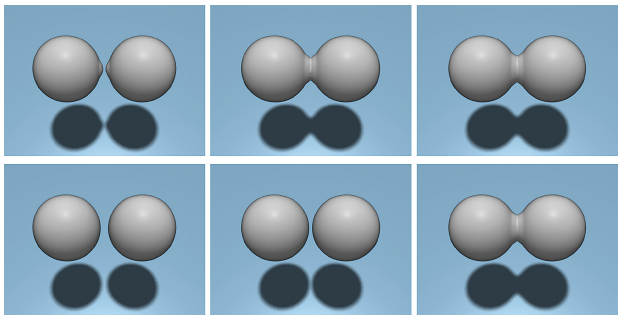


Fig. 4. Merging spheres. Top: Not accounting for components. Bottom: Correction using connected components.

be denoted by c_i. The definition of w_{ij} is then modified to be

$$w_{ij} = \begin{cases} 1 - (\|\mathbf{x}_i - \mathbf{x}_j\|)/r_i)^3 & \text{if } \|\mathbf{x}_i - \mathbf{x}_j\| < r_i \text{ and } c_i = c_j. \\ 0 & \text{otherwise} \end{cases} \tag{17}$$

With modified w_{ij}, the covariance matrix $\mathbf{C}_i$ and the anisotropic kernel $W(\mathbf{x} - \mathbf{x}_i, \mathbf{G}_i)$ is no longer affected by the neighbors that belong to different components. As a comparison between the original approach and the modified CC approach, Figure 4 shows two spheres approaching each other. As shown in the top row, the two spheres begin attracting each other before the collision when not using the CC labels. In contrast, the bottom row shows that the CC approach avoids this artifact and delays the merge due to the collision until the two spheres are much closer. This simulation used 23k particles. The overhead imposed by the CC computation is rather small. On average, the CC computation required 0.17 seconds per frame, while the surface reconstruction took 1.69 seconds per frame.

5. IMPLEMENTATION

We use the marching cubes algorithm [Lorensen and Cline 1987] to create a mesh that represents the fluid surface from the scalar field of Eq. (8). We represent obstacles as signed distance fields for collision detection with particles.

5.1 Singular Value Decomposition

In order to capture the anisotropy for each particle, a robust and efficient SVD algorithm needs to be implemented. In our 3D examples, we use Cardano's method [Smith 1961; Kopp 2008] to determine three singular values of the symmetric matrix $\mathbf{C}$ of Eq. (12). Although iterative methods such as two-sided Jacobi or QL are numerically more stable, Cardano's method is efficient because it analytically determines singular values, and it is robust enough for our approach because $\mathbf{C}$ is not ill-conditioned in most cases. From Cardano's method, we obtain three singular values $\sigma_1, \sigma_2, \sigma_3$ (Eq. (13). Then we use a cofactor matrix of $\mathbf{C} - \sigma_i \mathbf{I}$ to determine a corresponding normalized singular vector $\mathbf{v}_i$ [Carchid 1986].

Since the cofactor method works best for singular values of multiplicity one, we identify different cases based upon the multiplicity of the singular values. Let us denote the machine precision by ϵ, and define a relation $a \approx b$ to be true when $|a - b| < \epsilon a$. When $\sigma_1 \approx \sigma_3$ we assume that the three singular values are nearly identical, and we set the singular vectors to be the column vectors of an identity matrix. When $\sigma_1 \not\approx \sigma_2$ and $\sigma_2 \approx \sigma_3$, we compute a singular vector $\mathbf{v}_1$, and choose $\mathbf{v}_2$ and $\mathbf{v}_3$ as two arbitrary orthonormal vectors in the plane normal to $\mathbf{v}_1$. The other multiplicity two case is handled by exchanging σ_1 and σ_3 in the previous case. If the singular values do not match any of the previous cases, we assume that they all have multiplicity one. We first compute $\mathbf{v}_1$ and $\mathbf{v}_3$. In order to correct the numerical error, we make sure that $\mathbf{v}_1$ are $\mathbf{v}_3$ orthogonal by computing $\mathbf{v}_3 - \mathbf{v}_1(\mathbf{v}_1 \cdot \mathbf{v}_3)$ and using this as an updated version of $\mathbf{v}_3$. The remaining singular vector $\mathbf{v}_2$ is computed by the cross-product $\mathbf{v}_1 \times \mathbf{v}_3$.

5.2 Neighborhood Search

For neighborhood searches, we use a variation of the hash grid described in Adams and Wicke [2009] that handles the ellipsoidal support of our smoothing kernels. At every reconstruction step, we first compute an Axis Aligned Bounding Box (AABB) for the ellipsoid that is associated with each particle. Then we select the uniform grid cells that overlap with the AABB and retrieve the hash grid cells corresponding to these selected cells. We then store an index of the particle in these retrieved cells. In order to find the neighboring particles at an given point, we determine the hash grid cell that contains the point, examine the particles whose indices are stored in the cell, and tag as neighbors the ones whose ellipsoids contain the point.

5.3 Optimization of Performance

One bottleneck of our approach is that we perform SVD on all particles in order to stretch them. A simple performance optimization is to only create stretched particles near the surface of the fluid, and not in the fluid bulk, in order to accelerate the process of surface reconstruction. When simulation particles are regularly sampled, we use a simple criterion to isolate the near-surface particles: For a particle i, we count the number N of neighborhood particles within the smoothing radius r_i and compute the center of mass $\mathbf{m}$ of these particles. A particle i is tagged as near-surface particles if $|N - N_s| > 0.1N_s$ or $\|\mathbf{m} - \mathbf{x}_i\| > 0.1r_i$, where N_s is the number of neighborhood particles for an inner-volume particle with rest density. Once all of the particles have been examined, we perform SVD only on the tagged particles, while untagged particles remain unstretched using $\mathbf{G} = \mathbf{I}$.

Another bottleneck of our approach comes from the evaluation of the scalar field ϕ at the marching cubes grid points. In contrast to the isotropic kernel, the anisotropic kernel involves an extra matrix-vector multiplication $\|\mathbf{G}(\mathbf{x} - \mathbf{x}_i)\|$, and the scalar field value at one grid point is usually contributed to by multiple neighborhood kernels. In order to reduce this extra computational cost, we first introduce a fast exclusion test to check whether a grid point is out of the smoothing kernel support, formulated as $\|\mathbf{G}(\mathbf{x} - \mathbf{x}_i)\| > h_i$. Because $\mathbf{G}$ transforms a sphere into an ellipsoid, the smallest singular value σ_{min} of $\mathbf{G}$ determines a radius of the inner sphere bounded by an ellipsoid that is transformed from the unit sphere. Therefore $\sigma_{min}\|(\mathbf{x} - \mathbf{x}_i)\| > h_i$ is a sufficient condition for $\|\mathbf{G}(\mathbf{x} - \mathbf{x}_i)\| > h_i$, and a matrix-vector multiplication can be avoided when this condition holds. In addition to the exclusion test, we further reduce the computation cost by observing that the value of ϕ is close to zero near the surface and increases to one towards the inside of the fluid volume. Therefore, we can assume that a grid point at $\mathbf{x}$ is inside the volume and far away from the surface if $\phi(\mathbf{x}) > t$, where t is a positive threshold less than one. At each grid point, we start adding contributions from the nearby particles to the grid point's scalar value. Once the scalar value reaches t, we stop adding contributions for the remaining particles and use t as a scalar field value for the point. For our simulation examples, the value $t = 0.2$ is used. This technique eliminates many of the smoothing kernel evaluations for a given simulation. We refer readers Section 6 for the timings on the performance improvement.

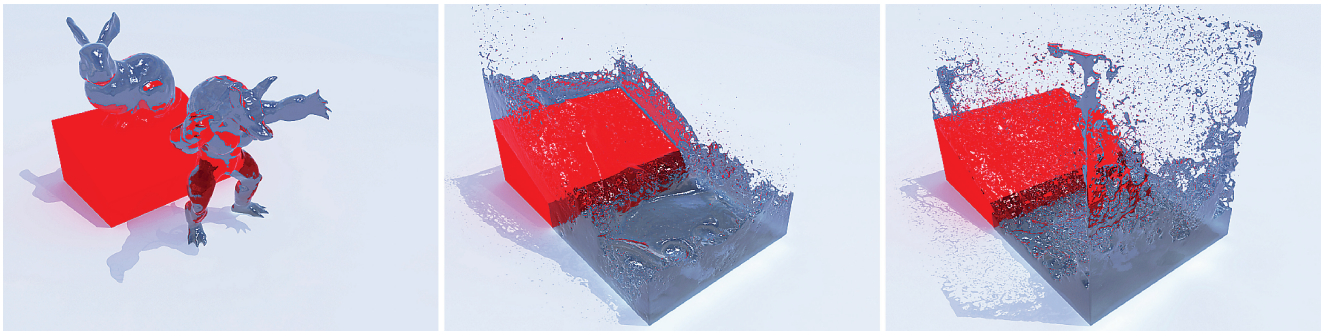


Fig. 5. Falling armadillo and bunny (one million particles).

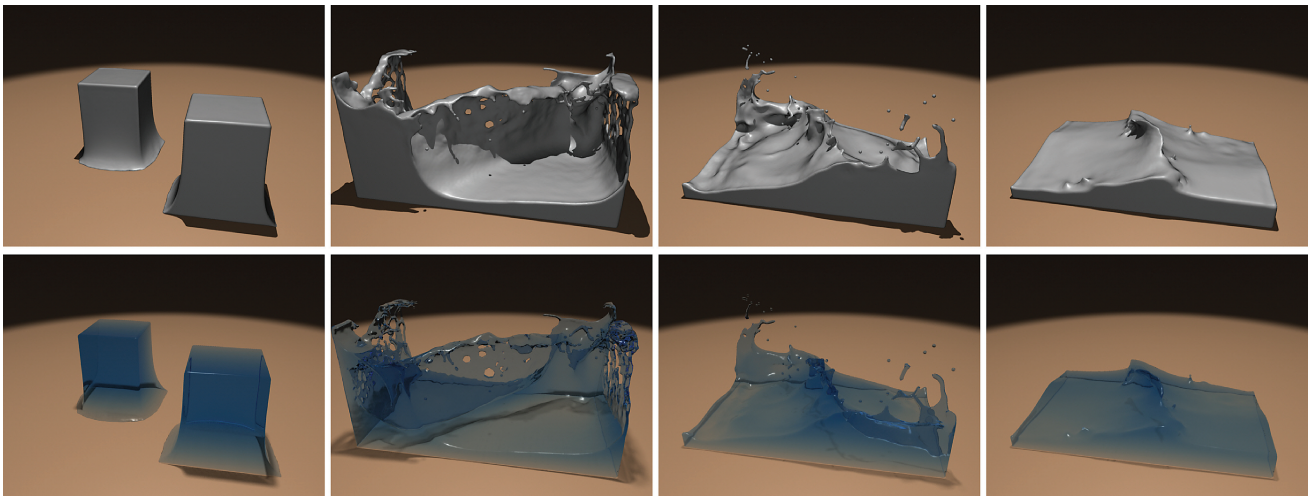


Fig. 6. Double dam break simulation.

6. RESULTS

In this section, we describe five simulations that were used to evaluate our surface reconstruction method: a water crown, flow on the bunny, a double dam break, falling figures, and agitated water. All of our surface reconstruction algorithms were run on a 2.4 GHz Intel Core2 Duo CPU with 1.72GB of memory. We use NVIDIA Gelato for rendering the resulting animations except the case of Figure 8 and Figure 5 which were rendered with Maya. All of our results but the agitated water were simulated using the Weakly Compressible SPH (WCSPH) approach of Becker and Teschner [2007]. We used a fixed time step of 0.0002s for running our simulations. The agitated water example was created by Robert Bridson at University of British Columbia using Naiad simulator. For the double dam break example, we present an interactive animation created by Simon Green at NVIDIA, who parallelized our method on the GPU.

Figure 1 shows an animation of a small drop of water that splashes into a larger body of water, causing a water crown. Only 24k particles were used to create this simulation. Note that even at this low particle count, the particle-based nature of the simulation is difficult to discern from the images. Our anisotropic kernel reconstruction of the surface creates a water crown that is smooth and unbroken near its base, and produces plausible pinch-off at the top. When the fluid rebounds in the center, a thin spike of water is maintained due to the stretching of the smoothing kernels along the spike's axis. When the water settles, the surface is smooth.

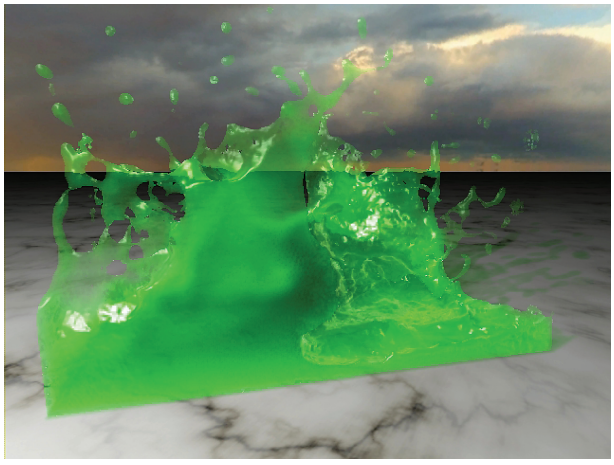


Fig. 7. Interactive double dam break (image courtesy of Simon Green).

Figure 3 shows a viscous fluid that is poured over the Stanford bunny. The fluid sheet that falls runs off the bunny is thin, usually

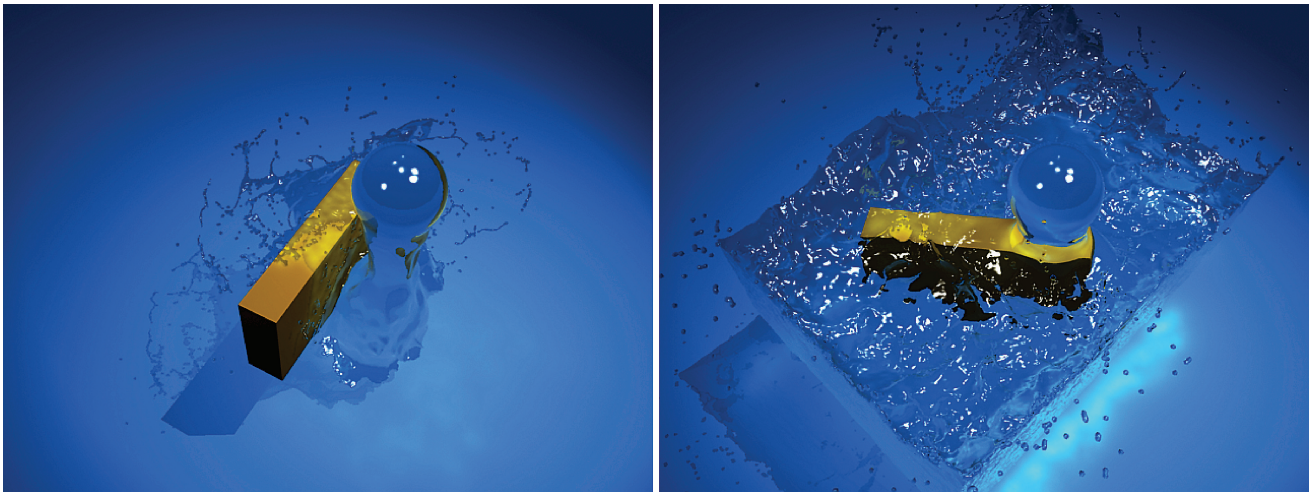


Fig. 8. This animation shows water spilling out of a spherical source and being agitated in a pool. FLIP was used to simulate this sequence, and it contains ten million particles. (This Naiad simulation data is courtesy of Robert Bridson and Exotic Matter AB.)

Table I. Average per Frame Timings (in minutes) and Speed Gains for Optimized Surface Reconstruction Methods on Two Examples

	Falling Figures	Agitated Water
Marching Cubes	0.59[3.09], 5.2x	5.40[23.90], 4.4x
Anisotropy	0.83[1.07], 1.3x	12.71[-], -
Total	1.42[4.16], 2.9x	18.11[36.61], 2.0x

The timings for unoptimized methods are shown in brackets.

just one particle thick, and yet the sheet is flat and smooth. In this sheet, the kernels are stretched in the two dimensions that run parallel to the sheet, and are compressed perpendicular to the sheet.

In the animation of Figure 5, we demonstrate the performance of our approach on a large-scale simulation with one million particles. Two fluid versions of the bunny and the armadillo fall down an inclined plane into a pool. In this example, our optimization techniques from Section 5.3 are used. Table I shows the improvement on the timings when these optimization techniques are applied.

Our next simulation is a double dam break (Figure 6), in which two blocks of water at opposite sides of a tank are suddenly released. The two parcels of water rush towards each other, collide in the center of the tank, and this throws up a thin sheet of water that runs diagonally across the tank. Similar to the bunny simulation, this thin sheet is often just one particle thick.

Figure 7 shows an interactive animation of the double dam break, using NVIDIA's CUDA as our parallel programming architecture. For the surface reconstruction, a CUDA kernel with a thread per particle is used for point denoising and anisotropic kernel computation. The fluid surfaces are rendered using ray-casting in a pixel shader instead of marching cubes. For each particle, only a bounding quad is rendered. The pixel shader calculates the intersection between the view ray and the ellipsoid (defined by the matrix), and also calculates the depth and surface normal. 128k particles were used for this example. The simulation, surface reconstruction, and rendering were simultaneously run on a GeForce GTX 460 video card and the interactive frame rate of 25fps was achieved. This example successfully demonstrates that parallelizing the surface creation process is simple and effective, since the approach only depends on local information about particle information.

In the animation of Figure 8, we show the result of our approach applied to an example that was simulated using FLIP. The fluid falls down from an emitter in the air into a pool, and it is agitated by an elongated block inside the pool. The simulation was run on a 2.3 GHz Intel Core i7 MacBook Pro laptop (4GB 1333 MHz DDR3 RAM). There were ten million particles used by the end of the simulation, and simulating each rendering frame took about 15 minutes per frame in worst case with 3.3 million grid voxels. A maximum of roughly 20 particles per voxel were allowed and the regularization was performed by deleting particles from overly full voxels. During emission, particles near the surface were also supersampled. This example demonstrates that our approach is not specific to SPH, but is also applicable to FLIP, which is widely used in production studios. Because particles near the surface were adaptively sampled during the emission, the simple optimization of isolating near-surface particles from Section 5.3 was not applicable for this example. However the other optimization to the evaluation of the scalar field was used, and the improvement in timings are shown in Table I.

6.1 Comparison and Limitations

Figure 9 shows a comparison between an isotropic surface reconstruction approach [Müller et al. 2003], Zhu and Bridson's approach [2005], the method of Adams et al. [2007], and our anisotropic kernel approach. The simulation that is used for comparison is the double dam break simulation with 140K particles. As the figure shows, the isotropic reconstruction method produces unacceptably bumpy surfaces. Zhu and Bridson's approach creates noticeably smoother surfaces, but some surface bumps are still apparent. In fairness to their method, Zhu and Bridson also perform a small amount of additional grid-based smoothing that we have omitted. The method of Adams et al. produces a still smoother surface, and this method creates the highest-quality surfaces from among the prior methods that we have tested. Our anisotropic kernel method produces surfaces that are even smoother than the method of Adams et al. Moreover, our method creates a thin sheet of water in the center of the image that is largely unbroken, where the method of Adams et al. creates a sheet with many holes in a lace-like pattern.

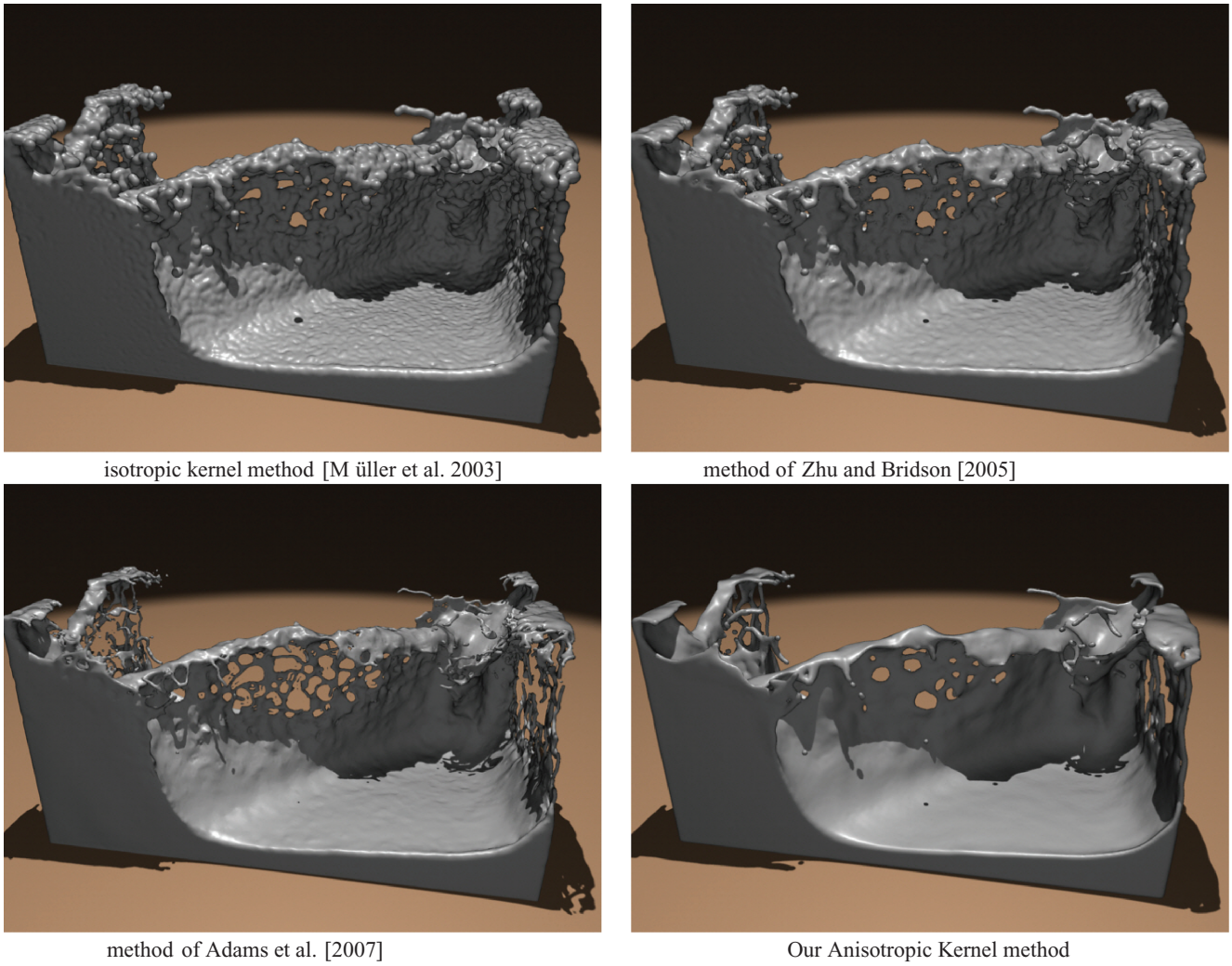


Fig. 9. Comparison between different surface reconstruction approaches on the Double dam break animation.

Table II. Average per Frame Timings (in minutes) for Four Surface Reconstruction Methods on the Double Dam Break Simulation

Reconstruction method	Surface reconstruction	Simulation	Opaque rendering	Transparent rendering
Isotropic	0.39	2.19	0.64	20.08
Zhu and Bridson	0.50			
Adams	1.76			
Anisotropic	0.96			

Table II shows timings for the double dam break example. The dimensions of the marching cubes grid for these results is $230 \times 190 \times 350$. The timings are per-frame averages (in minutes) across all of the frames of the animation. Our surface reconstruction approach is roughly twice as expensive as the isotropic kernel method and Zhu and Bridson's approach. The method of Adams et al. is the most time consuming of the four methods, due to the need to recalculate the signed distance field at a rate of 300 frames per second. When we dropped this recalculation to a lower rate, the surface results from the Adams method became noticeably lower in quality.

In Figure 10, we compare the effects of our approach's two main components: diffusion smoothing of particle positions and anisotropic kernels. The figure shows the result of using just one of these methods at a time. The top image shows a surface reconstructed from relocated particles after diffusion smoothing, but that uses only isotropic kernels. The middle image shows a surface reconstructed using anisotropic kernels, but without performing any smoothing of the particle positions. The bottom image shows the result of our full algorithm: a surface reconstructed from relocated particles and with anisotropic kernels. As the figure shows, the relocation step alone apparently reduces the surface bumps by

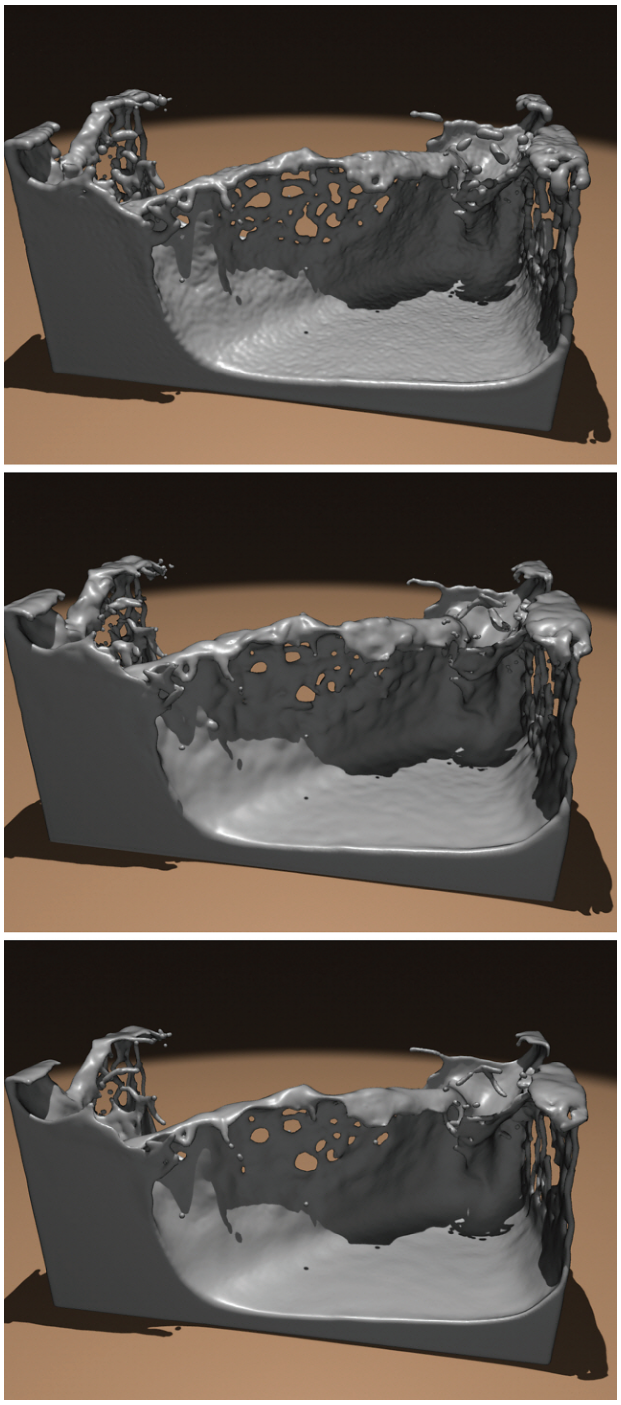


Fig. 10. Comparison between surface reconstruction variations. Top image shows smoothing of particle positions using isotropic kernels, middle shows use of anisotropic kernels but no positional smoothing, and bottom shows both anisotropic kernels and smoothing of the particle positions.

smoothing out irregular particle distributions, but small-scale noise is still visible due to the spherical shapes of the isotropic kernels. Even though unaltered particles with anisotropic kernels produce a smoother surface, the unsmoothed particle positions reveal the pattern of ellipsoidal particle shapes as ripples on the surface. This test demonstrates that our approach achieves the maximum quality in surface smoothness by combining diffusion smoothing and anisotropic kernels.

There are several limitations to our approach for surface reconstruction. Perhaps the most important caveat is that the surfaces that are created using this method contain less volume than prior approaches. This is due to the averaging of particle centers. Appendix A gives an analysis of this volume difference, and demonstrates that a particle near a flat surface will move a fraction of the smoothing kernel radius h_i. Unlike mesh-based smoothing approaches, however, our method does not shrink the surface near thin sheets of fluids. Also note that this smaller volume is only a side-effect of the surface reconstruction process, and it is not carried into the physics of the simulation.

Even though our method produces surfaces that have less noise than the other methods that we tested, it is still possible to see small bumps when the surface is magnified. These slight variations in the surface can be seen in the pattern of the caustics of the water crown animation when the water settles. We think that these slight ripples could easily be smoothed away using mesh-based smoothing, but we left our meshes unaltered in order to clarify what can be achieved using anisotropic kernels alone.

7. CONCLUSION AND FUTURE WORK

We have presented a new method of reconstructing surfaces from particle-based fluid simulations. This method relies on repositioning and stretching the kernels for each particle according to the local distribution of particles in the surrounding area. Our method preserves thin fluid sheets, maintains sharp features, and produces smooth surfaces when the simulated fluid settles. This method is also competitive in speed compared to other recent techniques for SPH surface reconstruction.

There are several avenues for future work using this method. One possibility is to smooth away the slight ripples on the reconstructed surface by using the smooth particle skinning method [Williams 2008] or its level-set improvement [Sin et al. 2009] on the ellipsoidal kernel representation. Another challenge is to apply our approach to particle level-set simulations [Enright et al. 2002]. Finally, it would be interesting to investigate whether there is a way to carry texture information along with the surface, as is possible using semi-Lagrangian contouring [Bargteil et al. 2006].

APPENDIX

A. ANALYSIS OF VOLUME SHRINKAGE

Our surface reconstruction approach applies a smoothing step to the positions of the kernel centers, and this pulls the kernels on the boundary towards the bulk of the fluid. Typically, this results in a slight amount of volume shrinkage when the fluid surface is reconstructed. In this appendix, we estimate the maximum distance between the original positions of the kernel centers and the smoothed positions of the kernels for particles that are on a flat surface. Suppose that a particle i is located at the origin of an Eulerian coordinate system and the neighboring particles are continuously located in a hemisphere of radius r_i above the xy plane. Due to the spatial symmetry, the weighted mean is on the positive z-axis. We compute the length of the weighted mean $\|\mathbf{x}_i\|$ from Eq. (10) and Eq. (11) using spherical coordinates. The

numerator and the denominator of $\|\mathbf{x}_i^w\|$ are formulated as

$$\|\sum_j w_{ij}\mathbf{x}_j\| = \pi \int_0^{r_i}\int_0^{\frac{\pi}{2}} \sin 2\phi(r^3 - r^6 r_i^{-3})drd\phi, \qquad (18)$$

and

$$\sum_j w_{ij} = 2\pi \int_0^{r_i}\int_0^{\frac{\pi}{2}} \sin\phi(r^2 - r^5 r_i^{-3})drd\phi. \qquad (19)$$

A simple algebraic manipulation yields

$$\|\mathbf{x}_i^w\| = 9r_i/28. \qquad (20)$$

Using Eq. (6), we estimate a bound between the kernel center position $\mathbf{x}_i$ and the updated center position $\overline{\mathbf{x}}_i$ after the volume smoothing by

$$\|\overline{\mathbf{x}}_i - \mathbf{x}_i\| = \|\lambda(\mathbf{x}_i^w - \mathbf{x}_i)\| \leq 9\lambda r_i/28. \qquad (21)$$

With the values $\lambda = 0.9$, $r_i = 2h_i$ that we used in our examples, we obtain $\|\overline{\mathbf{x}}_i - \mathbf{x}_i\| \leq 0.58h_i$. Since the extracted iso-surface encloses $\overline{\mathbf{x}}_i$'s, the maximal distance from the reconstructed surface to simulation particles on the surface is less than $0.58h_i$. This analysis shows that the volume shrinkage effect will not cause significant visual artifacts, since h_i is small compared to the size of simulation domain.

ACKNOWLEDGMENTS

We would like to thank Simon Green at NVIDIA for allowing us to use his interactive CUDA example in our article. We also thank Robert Bridson at University of British Columbia for providing his FLIP simulation data and giving us permission to use this data in our work. Finally, we would like to thank Chris Wojtan, Karthik Raveendran, Sehoon Ha, Joonho Baek, and the members of the Computer Graphics Lab in Georgia Institute of Technology for their help in simulation and rendering.

REFERENCES

ADAMS, B., PAULY, M., KEISER, R., AND GUIBAS, L. J. 2007. Adaptively sampled particle fluids. *ACM Trans. Graph. 26,* 3, 48.

ADAMS, B. AND WICKE, M. 2009. Meshless approximation methods and applications in physics based modeling and animation. In *Eurographics Tutorials*. 213–239.

BARGTEIL, A., GOKTEKIN, T., O'BRIEN, J., AND STRAIN, J. 2006. A semi-Lagrangian contouring method for fluid simulation. *ACM Trans. Graph. 25,* 1, 38.

BECKER, M., IHMSEN, M., AND TESCHNER, M. 2009a. Corrotated sph for deformable solids. In *Proceedings of the Eurographics Workshop on Natural Phenomena*.

BECKER, M. AND TESCHNER, M. 2007. Weakly compressible sph for free surface flows. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. Eurographics Association, 209–217.

BECKER, M., TESSENDORF, H., AND TESCHNER, M. 2009b. Direct forcing for Lagrangian rigid-fluid coupling. *IEEE Trans. Visual. Comput. Graph. 15,* 3, 493–503.

BHATTACHARYA, H., GAO, Y., AND BARGTEIL, A. W. 2011. A level-set method for skinning animated particle data. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*.

BLINN, J. 1982. A generalization of algebraic surface drawing. *ACM Trans. Graph. 1,* 3, 235–256.

BROCHU, T. AND BRIDSON, R. 2006. Fluid animation with explicit surface meshes. In *Proceedings of the Symposium on Computer Animation, Poster Session*.

BROCHU, T. AND BRIDSON, R. 2009. Robust topological operations for dynamic explicit surfaces. *SIAM J. Sci. Comput. 31,* 4, 2472–2493.

CARCHID, M. 1986. A method for finding the eigenvectors of an n n matrix corresponding to eigenvalues of multiplicity one. *Ameri. Math. Mont. 93,* 8, 647–649.

DESBRUN, M. AND CANI-GASCUEL, M. 1998. Active implicit surface for animation. In *Proceedings of the Graphics Interface*. 143–150.

DINH, H., TURK, G., AND SLABAUGH, G. 2001. Reconstructing surfaces using anisotropic basis functions. In *Proceedings of the International Conference on Computer Vision (ICCV)*. Vol. 2. 606–613.

ENRIGHT, D., FEDKIW, R., FERZIGER, J., AND MITCHELL, I. 2002. A hybrid particle level set method for improved interface capturing. *J. Comput. Phys. 183,* 1, 83–116.

ENRIGHT, D., LOSASSO, F., AND FEDKIW, R. 2005. A fast and accurate semi-Lagrangian particle level set method. *Comput. Struc. 83,* 6-7, 479–490.

ENRIGHT, D., MARSCHNER, S., AND FEDKIW, R. 2002b. Animation and rendering of complex water surfaces. *ACM Trans. Graph. 21,* 3, 736–744.

HIRT, C. AND NICHOLS, B. 1981. Volume of fluid/VOF/ method for the dynamics of free boundaries. *J. Comput. Phys. 39,* 1, 201–225.

KALAIAH, A. AND VARSHNEY, A. 2003. Statistical point geometry. In *Proceedings of the Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*. Eurographics Association, 115.

KEISER, R., ADAMS, B., GASSER, D., BAZZI, P., DUTRE, P., AND GROSS, M. 2005. A unified Lagrangian approach to solid-fluid animation. In *Proceedings of the Eurographics/IEEE VGTC Symposium Point-Based Graphics 0*, 125–148.

KOPP, J. 2008. Efficient numerical digonalization of hermitian 3 × 3 matrices. *Inte. J. Modern Phys. C 19,* 03, 523–548.

KOREN, Y. AND CARMEL, L. 2003. Visualization of labeled data using linear transformations. In *Proceedings of IEEE Information Visualization*. Vol. 2003.

LENAERTS, T., ADAMS, B., AND DUTRÉ, P. 2008. Porous flow in particle-based fluid simulations. In *Proceedings of the ACM SIGGRAPH Papers*. ACM, New York, 1–8.

LIU, M. B., LIU, G. R., AND LAM, K. Y. 2006. Adaptive smoothed particle hydrodynamics for high strain hydrodynamics with material strength. *Shock Waves 15*, 21–29.

LORENSEN, W. E. AND CLINE, H. E. 1987. Marching cubes: A high resolution 3d surface construction algorithm. In *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*. ACM, New York, 163–169.

MONAGHAN, J. 1994. Simulating free surface flows with sph. *J. Comput. Phys. 110,* 2, 399–406.

MÜLLER, M. 2009. Fast and robust tracking of fluid surfaces. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. ACM, 237–245.

MÜLLER, M., CHARYPAR, D., AND GROSS, M. 2003. Particle-based fluid simulation for interactive applications. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. Eurographics Association, 154–159.

MÜLLER, M. AND CHENTANEZ, N. 2011. Solid simulation with oriented particles. In *ACM SIGGRAPH Papers (SIGGRAPH '11)*. ACM, New York, 92:1–92:10.

MÜLLER, M., KEISER, R., NEALEN, A., PAULY, M., GROSS, M., AND ALEXA, M. 2004a. Point based animation of elastic, plastic and melting objects. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. Eurographics Association, 141–151.

MÜLLER, M., SCHIRM, S., TESCHNER, M., HEIDELBERGER, B., AND GROSS, M. 2004b. Interaction of fluids with deformable solids. *J. Comput. Anim. Virt. Worlds*. 159–171.

MÜLLER, M., SOLENTHALER, B., KEISER, R., AND GROSS, M. 2005. Particle-Based fluid-fluid interaction. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. ACM, New York, 237–244.

MUSETH, K., CLIVE, M., AND ZAFAR, N. B. 2007. Blobtacular: Surfacing particle system in "Pirates of the Caribbean 3". In *SIGGRAPH Sketches*.

OSHER, S. AND FEDKIW, R. 2002. *Level Set Methods and Dynamic Implicit Surfaces*. Springer.

OWEN, J. M., VILLUMSEN, J. V., SHAPIRO, P. R., AND MARTEL, H. 1995. Adaptive smoothed particle hydrodynamics: Methodology II *Astrophys. J. 116*, 155.

PREMOZE, S., TASDIZEN, T., BIGLER, J., LEFOHN, A., AND WHITAKER, R. T. 2003. Particle-based simulation of fluids. In *Proceedings of Eurographics Conference*. 401–410.

SIN, F., BARGTEIL, A., AND HODGINS, J. 2009. A point-based method for animating incompressible flow. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*.

SMITH, O. 1961. Eigenvalues of a symmetric 3×3 matrix. *Comm. ACM 4,* 4, 168.

SOLENTHALER, B. AND PAJAROLA, R. 2008. Density contrast sph interfaces. In *Proceedings of ACM SIGGRAPH / EG Symposium on Computer Animation*. 211–218.

SOLENTHALER, B., SCHLÄFLI, J., AND PAJAROLA, R. 2007. A unified particle model for fluid–solid interactions: Research articles. *Comput. Animat. Virtual Worlds 18,* 1, 69–82.

STRAIN, J. 2001. A fast semi-Lagrangian contouring method for moving interfaces. *J. Comput. Phys. 170,* 1, 373–394.

TAUBIN, G. 2000. Geometric signal processing on polygonal meshes. In *Eurographics State of the Art Reports*.

THÜREY, N., KEISER, R., PAULY, M., AND RÜDE, U. 2006. Detail-preserving fluid control. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. Eurographics Association. 7–12.

WILLIAMS, B. W. 2008. Fluid surface reconstruction from particles. M.S. thesis, The University of British Columbia, Canada.

WOJTAN, C. 2011. Liquid simulation with mesh-based surface tracking. In *ACM SIGGRAPH Courses (SIGGRAPH '11)*. ACM, New York, 8:1–8:84.

WOJTAN, C., THÜREY, N., GROSS, M., AND TURK, G. 2009. Deforming meshes that split and merge. In *Proceedings of the ACM SIGGRAPH Papers*. ACM, 76.

WOJTAN, C., MULLER-FISCHER, M., AND BROCHU, T. 2011. Liquid simulation with mesh-based surface tracking. In *ACM SIGGRAPH '11 Courses*. ACM.

YU, J. AND TURK, G. 2010. Reconstructing surfaces of particle-based fluids using anisotropic kernels. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. Eurographics Association, 217–225.

ZHU, Y. AND BRIDSON, R. 2005. Animating sand as a fluid. In *Proceedings of the ACM SIGGRAPH Papers*. ACM, 972.

Received August 2011; revised April 2012; accepted May 2012