On the Velocity of an Implicit Surface

JOS STAM and RYAN SCHMIDT
Autodesk Research

In this paper we derive an equation for the velocity of an arbitrary time-evolving implicit surface. Strictly speaking only the normal component of the velocity is unambiguously defined. This is because an implicit surface does not have a unique parametrization. However, by enforcing a constraint on the evolution of the normal field we obtain a unique tangential component. We apply our formulas to surface tracking and to the problem of computing velocity vectors of a motion blurred blobby surface. Other possible applications are mentioned at the end of the paper.

Categories and Subject Descriptors: I.3.5 [**Computer Graphics**]: Computational Geometry and Object Modeling; I.3.6 [**Computer Graphics**]: Methodology and Techniques

General Terms: Implicit surfaces, Blobbies, motion blur

Additional Key Words and Phrases: Implicit surfaces, Blobbies, motion blur

ACM Reference Format:

Stam, J., and Schmidt, R. 2011. On the Velocity of an Implicit Surface. ACM Trans. Graph. XX, Y, Article ZZZ (Month 2010), NN pages.
DOI = 10.1145/1559755.1559763
http://doi.acm.org/10.1145/1559755.1559763

1. INTRODUCTION

Implicit surfaces are defined as the iso-contour of a smooth function. This paper addresses what happens to the surface when that function varies over time. Specifically we are interested in the velocities of the points on the surface. Strictly speaking it only makes sense to talk about the normal component of the velocity along the gradient of the implicit function. This is because implicit surfaces admit many parametrizations which are not fixed by the implicit function. To understand this imagine rotating all the points lying on an implicit sphere by a fixed amount as shown in Figure 1. Each point will have a tangential velocity despite the fact that the sphere appears at rest. However, intuitively we know that if we sample the implicit surface, these points will have a definite velocity. For example, if we translate a sphere by a constant velocity, then each point of the surface will also move at this velocity. To fix the tangential velocity we need to impose another condition on the motion of these particles. In this paper we propose that the normalized gradient field should not change over time. This is indeed the case when the implicit surface undergoes a translational motion.

This work was initially motivated by the problem of motion blurring iso-surface meshes of a particle simulation. However, in estimating the tangential velocity we have also addressed the more general problem of predicting where a point on a time-varying implicit surface will move to in the next frame. With this building block we can more accurately track an animated implicit surface with a mesh or set of particles, rather than generating a new mesh at every frame. Similarly, surface properties like color or texture coordinates can be more easily and accurately propagated, improving frame coherence. We work out all the mathematical expressions for the surface velocity of a blobby surface, and show applications to surface tracking and motion blur. These results demonstrate that although our formula assumes translational motion, with moderate timesteps it performs well for other motion types.

2. RELATED WORK

We are not aware of any previous work addressing the problem of defining tangential components of surface velocity fields for time-varying implicit surfaces. A condition on the evolution of the normal field similar to ours was used by Mullan et al. to model implicit surfaces using radial basis functions [Mullan et al. 2004].

One of the primary benefits of an accurate velocity field is that it greatly simplifies the task of *tracking* the implicit surface with particles. In [Smets-Solanes 1996] a velocity field is explicitely specified along with the implicit surface. Surface tracking is common practice in level set simulation [Enright et al. 2002], where an existing velocity field drives the simulation. If a velocity field is not known a priori, the state-of-the-art approach [Witkin and Heckbert 1994; Rodrian and Moock 1996] is to use the well-known normal velocity at each particle. If the underlying motion has a tangential component, then under normal flow the particles will become unevenly distributed, and so some geometric energy must also be minimized to redistribute the particles. As normal flow can rapidly introduce large variations in sampling density, the cost of robustly minimizing these nonlinear energies is significant [Meyer et al. 2007]. With a tangential velocity estimate, the particles will more accurately track the actual surface motion and significantly less "massaging" will be necessary to ensure adequate particle distribution.

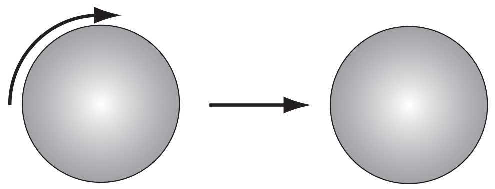
Fig. 1. Rotating the sphere does not change its implicit shape.

Authors' addresses: J. Stam and R. Schmidt, Autodesk, Inc., 210 King Street East, Toronto, Ontario, Canada, M5A 1J7; e-mail: Jos.Stam@autodesk.com, rms@dgp.toronto.edu.

If the particles are connected with mesh topology, the edge graph must be adapted to deal with any topological splits and merges, as well as to handle degeneracies, foldovers, and so on [Bouthors and Nesme 2007; Brochu and Bridson 2009]. These issues are outside the scope of our work, but we do note that improvements in particle tracking generally reduce the number of “events” that the mesh adaptation algorithm needs to handle.

3. NORMAL VELOCITY

A time evolving implicit surface is defined by a function F whose parameters change over time. More formally the surface is defined by the following set:

$$\Gamma(t) = \{\mathbf{x} | F(\mathbf{x}, t) = 0\}.$$

Information on the velocity of the points on the surface can be obtained by computing the time derivative of the implicit function:

$$\dot{F}(\mathbf{x}, t) = \frac{\partial F}{\partial t}(\mathbf{x}, t) + \nabla F(\mathbf{x}, t)^T \dot{\mathbf{x}}(t).$$

For points on the surface this derivative has to be zero. Therefore we have that:

$$\nabla F(\mathbf{x}, t)^T \dot{\mathbf{x}}(t) = -\frac{\partial F}{\partial t}(\mathbf{x}, t).$$

Or more succinctly:

$$\mathbf{q}^T \dot{\mathbf{x}}(t) = u.$$

This equation can be solved using the pseudo inverse (see [Moore 1920]) of $\mathbf{q}$, namely $\mathbf{q}^+ = \mathbf{q}^T / \mathbf{q}^T \mathbf{q}$, so that:

$$\dot{\mathbf{x}}(t) = \frac{u}{\mathbf{q}^T \mathbf{q}} \, \mathbf{q} = v_n \, \mathbf{n}, \tag{1}$$

where $\mathbf{n} = \mathbf{q}/|\mathbf{q}|$ is the normalized gradient and

$$v_n = \frac{u}{|\mathbf{q}|}.$$

Equation 1 provides only partial information on the velocity, only its variation in the direction of the gradient $\mathbf{q}$. The tangential component can be arbitrary since motion within the surface does not change the shape. This reflects the fact that an implicit surface does not have a unique parametrization.

4. TANGENTIAL VELOCITY

One obvious choice for the tangent velocity is to simply set it to zero. However, this doesn’t work in the example of a sphere moving at constant speed. As shown in Figure 2 the velocity field at the surface does not match the constant velocity, only for points whose normal is aligned with the velocity. This does not mean that the points leave the surface, it means that they will be bunched up at the poles. In general this is not desirable. What we propose in this paper is to require that the normal at each point does not vary over time. For translational motions this is clearly the case. Again we emphasize that in theory any tangent can be chosen. Our goal is to chose one that is close to the hypothetical motion of a surface particle and which is exact for translational motions.

Mathematically our condition states that the time derivative of the normalized gradient should vanish on the surface:

$$\frac{d}{dt}\left(\frac{\nabla F}{|\nabla F|}\right) = 0. \tag{2}$$

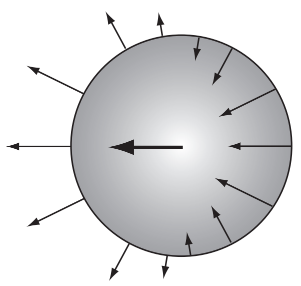

Fig. 2. Normal Velocity of a translating sphere.

To compute this derivative we use the fact that the derivative “D” of a normalized vector $\mathbf{n} = \nabla F / |\nabla F|$ is the projection of the derivative onto the plane normal to the vector. More precisely:

$$D \, \mathbf{n} = \frac{1}{|\nabla F|} \, \mathbf{P_n} \, D \, \nabla F, \tag{3}$$

where $\mathbf{P_n}$ is the projection operator to the plane normal to the vector $\mathbf{n}$:

$$\mathbf{P_n} = \mathbf{I}_3 - \mathbf{n}\mathbf{n}^T,$$

where $\mathbf{I}_3$ is the 3×3 identity matrix. Similar to the case of the normal velocity, we have that the (total) time derivative is equal to two terms:

$$\frac{d}{dt}\left(\frac{\nabla F}{|\nabla F|}\right) = \frac{\partial}{\partial t}\left(\frac{\nabla F}{|\nabla F|}\right) + \nabla \left(\frac{\nabla F}{|\nabla F|}\right) \dot{\mathbf{x}}.$$

Using Eq. 3 we can compute the two terms in this equation as follows:

$$\begin{aligned} \frac{\partial}{\partial t}\left(\frac{\nabla F}{|\nabla F|}\right) &= \frac{1}{|\nabla F|} \, \mathbf{P_n} \, \frac{\partial}{\partial t} \nabla F \\ \nabla \left(\frac{\nabla F}{|\nabla F|}\right) &= \frac{1}{|\nabla F|} \, \mathbf{P_n} \, \nabla \nabla^T F. \end{aligned}$$

Using these results we have that:

$$\frac{d}{dt}\left(\frac{\nabla F}{|\nabla F|}\right) = \frac{1}{|\nabla F|} \, \mathbf{P_n} \left(\frac{\partial}{\partial t} \nabla F + \nabla \nabla^T F \, \dot{\mathbf{x}}\right).$$

So that solving Eq. 2 gives the following equation for the tangential component of the velocity:

$$\mathbf{P_n} \mathbf{H}_F \, \dot{\mathbf{x}} = -\mathbf{P_n} \, \frac{\partial}{\partial t} \nabla F, \tag{4}$$

where $\mathbf{H}_F = \nabla \nabla^T F$ is the Hessian of the implicit function F.

5. EXAMPLE: MOVING BLOBS

In this section we use the derived formulas to compute the velocity of a blobby surface whose motion is given by the velocity of the center of each blob. These surfaces were first introduced in [Blinn 1982]. They are a popular way to visualize particle systems such as the output of a liquid simulator. More precisely the implicit field is defined as follows:

$$F(\mathbf{x}, t) = \sum_{i=1}^{n} w_i \; f(|\mathbf{x} - \mathbf{x}_i(t)| / \sigma_i) - T,$$

where the w_i are weights, the σ_i are radii, T is a threshold value and the function $f(r)$ is a smooth function equal to one at the origin and zero at one and beyond. For example one can use:

$$f(r) = (1-r^2)^3 \quad \text{and} \quad f'(r) = -6(1-r^2)^2\, r.$$

We have:

$$\begin{aligned} u &= -\frac{\partial F}{\partial t}(\mathbf{x}, t) \\ &= -6\sum_{i=1}^{n} \frac{w_i}{\sigma_i}\,(1-r_i^2)^2\, r_i\, \frac{(\mathbf{x}-\mathbf{x}_i(t))^T}{|\mathbf{x}-\mathbf{x}_i(t)|}\dot{\mathbf{x}}_i(t) \\ &= -6\sum_{i=1}^{n} \frac{w_i}{\sigma_i^2}\,(1-r_i^2)^2(\mathbf{x}-\mathbf{x}_i(t))^T\dot{\mathbf{x}}_i(t), \end{aligned}$$

where $r_i = |\mathbf{x}-\mathbf{x}_i(t)|/\sigma_i$. Similarly,

$$\mathbf{q} = \nabla F(\mathbf{x}, t) = -6\sum_{i=1}^{n} \frac{w_i}{\sigma_i^2}\,(1-r_i^2)^2\,(\mathbf{x}-\mathbf{x}_i(t)).$$

We now have all the ingredients to compute the normal velocity (Eq. 1).

Now let's work out the math for the tangential velocity. In the calculations below we always drop any expressions that are in the direction of $\mathbf{n}$ as they will be projected out, to simplify a lot of expressions. First we have that

$$\mathbf{P_n H}_F = -\mathbf{P_n}\, 6\sum_{i=1}^{n} \frac{w_i}{\sigma_i^2}\,(1-r_i^2)^2\,\mathbf{I}_3.$$

Similarly,

$$\mathbf{P_n}\,\frac{\partial}{\partial t}\nabla F = \mathbf{P_n}\, 6\sum_{i=1}^{n} \frac{w_i}{\sigma_i^2}\,(1-r_i^2)^2\,\dot{\mathbf{x}}_i.$$

And therefore we get that:

$$\mathbf{P_n}\dot{\mathbf{x}} = \mathbf{P_n}\,\frac{1}{W}\sum_{i=1}^{n} \frac{w_i}{\sigma_i^2}\,(1-r_i^2)^2\,\dot{\mathbf{x}}_i,$$

where

$$W = \sum_{i=1}^{n} \frac{w_i}{\sigma_i^2}\,(1-r_i^2)^2.$$

The tangent velocity is thus the weighted average of the projected velocities at the blob centers. However, in general the velocity (including the normal component) is not equal to the a simple weighting of the blob's velocities, which would seem like an obvious solution. To see that this is a poor choice consider the case of two blobs moving towards each other as shown in Figure 3. A simple weighting of the blob's velocities would result in a zero surface velocity in the center. However, clearly this velocity is non-zero as the surface is expanding there when the two blobs move towards each other. Our approach on the other hand predicts a more accurate velocity as we only use the weighted average for the tangential velocity component, while Eq. 1 gives us the non-zero normal velocity in the vertical direction.

6. EVALUATION

We evaluated our technique on two standard problems where blobby objects are used to visualize Lagrangian simulations: surface tracking and motion blur.

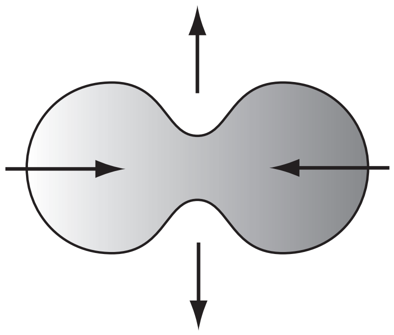
Fig. 3. Two blobs moving towards each other.

6.1 Surface Tracking

To integrate an animated implicit surface into a standard rendering pipeline, it must be converted into a mesh representation. This can be done by generating an entirely new mesh at each frame, for example using marching cubes [Wyvill et al. 1986; Lorensen and Kline 1987], or by generating a high-quality mesh at the first frame, and then at each frame updating the mesh to follow or *track* the implicit surface as best as possible [Bouthors and Nesme 2007].

Tracking a dynamic implicit surface with a mesh is a challenge because implicit surfaces are most often used when the surface topology will frequently change during the simulation. Recent techniques such as the *el Topo* tracker [Brochu and Bridson 2009] demonstrate that robust mesh-based implicit surface tracking is feasible even in very challenging cases. However, such techniques depend on the velocity field of the underlying implicit surface. In this section, we compare standard normal velocity with our improved *total* velocity approach, which also takes tangential velocity into account. Our results demonstrate that total velocity provides more robust and predictable results for surface tracking problems.

To generate the initial high-quality mesh, we perform a marching-cubes triangulation of the surface, followed by iterations of uniform Laplacian fairing and a re-projection step:

$$\mathbf{x} \leftarrow \mathbf{x} - \frac{F(\mathbf{x})\nabla F(\mathbf{x})}{\|\nabla F(\mathbf{x})\|^2} \tag{5}$$

to improve the vertex distribution [Ohtake et al. 2003].

6.1.1 *Basic Surface Motions.* We begin with the most basic tracking technique - at each frame, for each particle, we compute either the normal or total velocity and then apply the same forward Euler step as is used in the simulation that animates the blobs. The benefit of our total velocity is visible even in the simplest of cases, such as translation with uniform velocity. Under normal velocity,

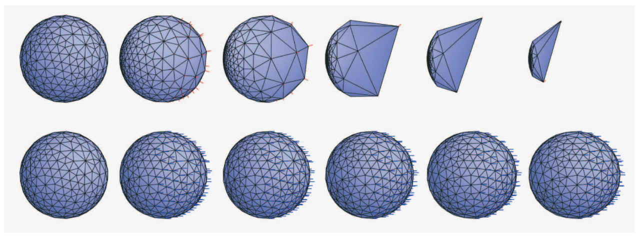
Fig. 4. An implicit blob moving to the right with uniform velocity is tracked using a mesh generated at the first frame (left). Top: normal velocity. Bottom: our total velocity approach.

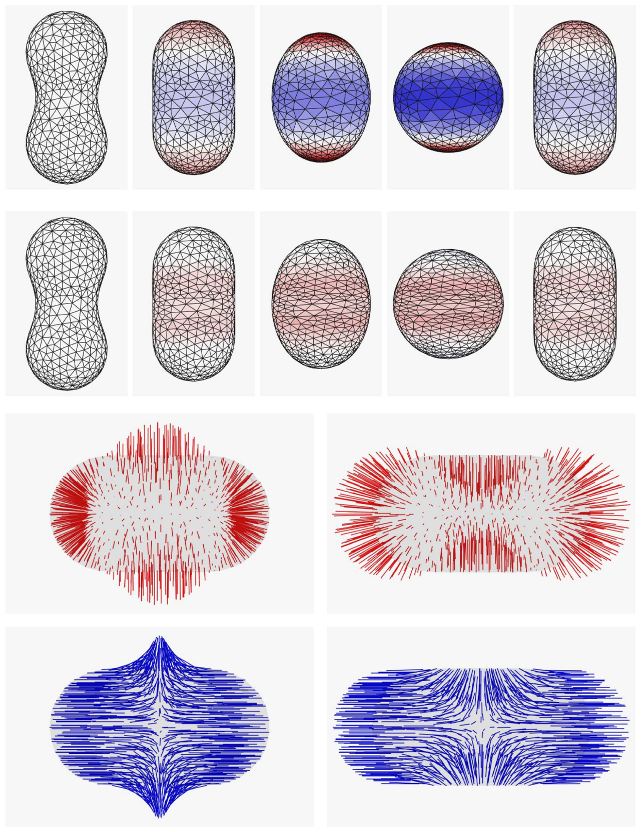

Fig. 5. A deforming implicit surface generated by two blended blobs connected with a spring is tracked using a mesh generated at the first frame (left). Triangles are colored based on area change, with blue and red indicating increased and decreased area, respectively. Normal (red) and total (blue) vertex velocities at second and fifth frame are also shown. Rows 1,3: normal velocity. Rows 2,4: our total velocity approach.

the mesh appears to "slide" over the surface and bunch up at critical points, while the total velocity result exactly tracks the underlying sphere (Figure 4).

Since our formulation was specifically derived to preserve rigidity of the normal field, exact results under translational motion are perhaps not surprising. It is less obvious whether our approach is appropriate for non-rigid motions. We examine such a case in Figure 5, where a "peanut" surface deforms as the distance between the two underlying blobs is varied. As this case is symmetric about the origin, tracking with normal velocity is stable, but the mesh undergoes significantly more deformation than with total velocity. The "endcaps" of the peanut, which only undergo translational motion, are deformed under normal velocity but remain rigid with total velocity. In the interior region, the surface is non-rigidly deforming, however the velocity field images clearly show that our total velocity provides more accurate estimates of the surface motion.

Another interesting test case is that of rotation, as our tangential velocity formulation does not explicitly incorporate rotational velocity (Sec 4). We consider the basic case of a blob orbiting around the origin in Figure 6. Again, we see that under normal velocity the surface quickly collapses, while total velocity is stable.

A standard method to improve surface tracking is to refine the mesh after the velocity update. Several techniques have been described to maintain a good distribution of particles on an implicit surface [Witkin and Heckbert 1994; Meyer et al. 2007], however these techniques assume that the particles can move arbitrarily. This quickly leads to triangle foldovers if applied to the vertices of a mesh. Instead, we store the edge lengths of the initial mesh. Then at each frame, for each vertex, we project the one-ring neighbourhood into the vertex tangent plane, add springs with the stored edge lengths as rest lengths, and then fix the neighbour positions and solve for the vertex using a simple mass-spring simulation. As with the velocity step, after each round of fairing we project the vertices back onto the surface using Equation 5.

Applying even one round of fairing at each frame stabilizes normal velocity tracking in the point-orbit test, while leaving the total velocity result relatively unchanged. If we measure the triangle area distortion, we observe that the mesh is still significantly distorted when using normal velocity. As we increase the number of fairing rounds, the normal velocity result improves, however this fairing has a significant cost; in this case, an order of magnitude more computation time per-frame is necessary to achieve the level of error generated by total velocity without fairing. The number of fairing rounds needed to prevent collapse is also speed/timestep-dependent. Similarly, if we increase the speed of rotation or take larger timesteps, additional fairing steps must be added to maintain stability for normal velocity tracking. With total velocity tracking, we observe only a small increase in area distortion.

6.1.2 *Particle-Based Tracking.* An alternative to tracking an implicit surface with a mesh is to discard the connectivity and track only with particles, which can then be rendered as surfels or splats [Witkin and Heckbert 1994; Levet et al. 2007]. This neatly handles the problems associated with topological changes in the underlying implicit surface. The trade-off is that to maintain a good distribution of particles, one must solve a complex global optimization problem which also requires maintenance of a spatial data structure. If the particle motion is determined by normal velocity, then this particle re-distribution must be performed each frame, otherwise the particles quickly bunch up and slide off the surface.

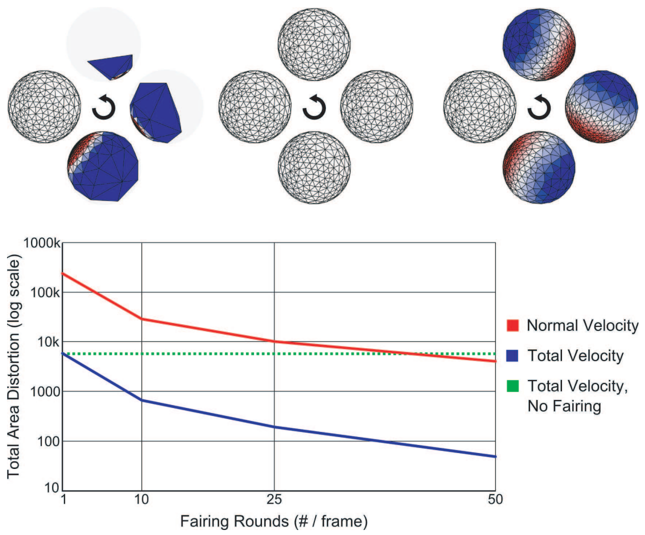

Fig. 6. Tracking of an implicit blob rotating around the origin can be made more robust by adding fairing steps at each frame. Top left: normal velocity. Top middle: our total velocity. Top right: normal velocity with one fairing round. Bottom: plot of mesh distortion with increasing rounds of fairing.

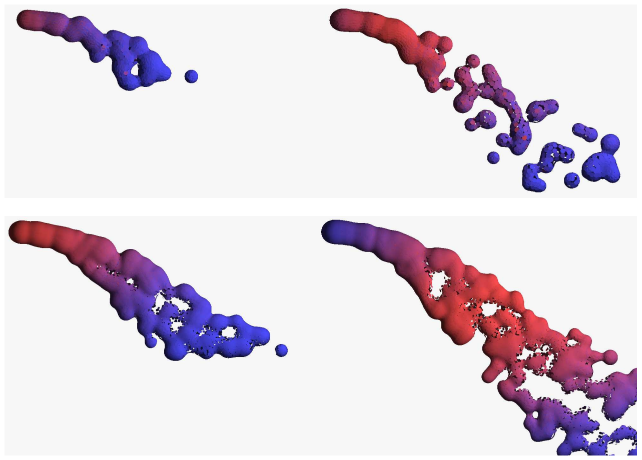
Fig. 7. Particle-based tracking of a basic simulation with time-varying color. Top: Total velocity tracking with per-frame particle re-distribution. Bottom: Total velocity, no re-distribution.

In Figure 7 we show particle-tracking results on a basic simulation. Blobs are generated at an emitter with a constant randomly-perturbed velocity, and then fall under gravity. When a blob is emitted, we also generate new particles at the emitter and project them to the surface. Figure 7,top shows a result which incorporates the particle redistribution of Witkin & Heckbert [1994]. We did not include particle splitting, so gaps inevitably appear as the blobs spread out, however the surface remains well-covered. Figure 7,bottom shows another run with higher blob and particle density, but with the redistribution disabled. This example makes it clear that the redistribution need only fill in the small undersampled regions that appear as the blobs pull apart; the tracking itself is highly robust. We note that similar simulations run with normal-velocity-based tracking quickly broke down.

Undersampling of the sort seen in Figure 7 can often be resolved simply by throwing more particles at the problem, although this does adversely affect computation times when each particle moves at every frame. However, in interactive modeling applications generally only a portion of the scene is being manipulated at once, so much higher particle densities can be used, allowing complex deformations and topological changes to be handled without expensive particle re-distribution. For example, in Figure 8 we interactively dragged a blob in a small loop, causing multiple topological changes. With normal velocity, the particles quickly spread and bunch up, leaving large gaps. Our total velocity exhibits much better performance. A few small holes do appear near the end of the manipulation, so occasional particle re-distribution would still be desirable, but can safely be deferred to idle time without affecting the ability of the user to understand the surface.

6.1.3 *Rigidity.* So far, we have focused on how well the tracked surface approximates the underlying implicit surface over time. Another desirable property is that the tracked points remain in consistent locations on the surface. Again, as the implicit surface lacks a natural parameterization, there is no well-defined notion of "surface location". However, in most cases we have some intuition for how a particle should move between frames.

In Figure 9 we assign a color to each tracked point based on the underlying implicit scalar field. These colors allow us to visualize how the tracked points "flow" over the surface during the simulation. As the shape is simply rotated about a central point, the ideal result is one in which the colors remain completely static. With normal velocity, the mesh slides across the surface extensively, to the point where the initial colors have been shifted counter-clockwise an entire blob after one revolution, while with total velocity we see a much smaller shift in the spatial location of the colored regions. In both cases, per-frame fairing is applied.

6.1.4 *Discontinuities.* So far we have focused on isosurfaces of smooth scalar fields, which are naturally produced by blended blobs. However, in many other implicit scenarios the underlying scalar fields have discontinuities. For example, combining two sets of blended blobs F_1 and F_2 with the $\max$ operation results in the Boolean *union* of their surfaces. The discontinuity in the $\max(F_1, F_2)$ operator becomes a discontinuity in the scalar field, producing a crease between the two surfaces. Along this contour the gradient and Hessian are mathematically undefined, and Eq. 4 cannot be evaluated. However, in practice $\nabla \max(F_1, F_2)$ is implemented by evaluating the values and gradients of each input field and selecting one to return, so the discontinuity is simply transferred to the velocity field.

Discontinuities in the velocity field are not a significant problem for even our simplistic tracking methods. In Figure 10 the implicit surface is the union of a rotating "stick" of blended blobs, and a single larger blob. With faired mesh-based tracking we see no more accumulated error than in experiments with smooth velocity fields. Note that the fairing steps do induce the blue portion of the mesh to rotate with the green and red regions, even though it has no velocity. In a particle tracker without fairing, red and green particles which slide across the crease immediately become stationary, while blue particles are "picked up" by the rotating stick. This leaves some undersampled regions, but at no point do the discontinuities cause the tracking to become unstable. Note, however, that field discontinuities may be more problematic if the gradient or Hessian is numerically approximated via finite differences.

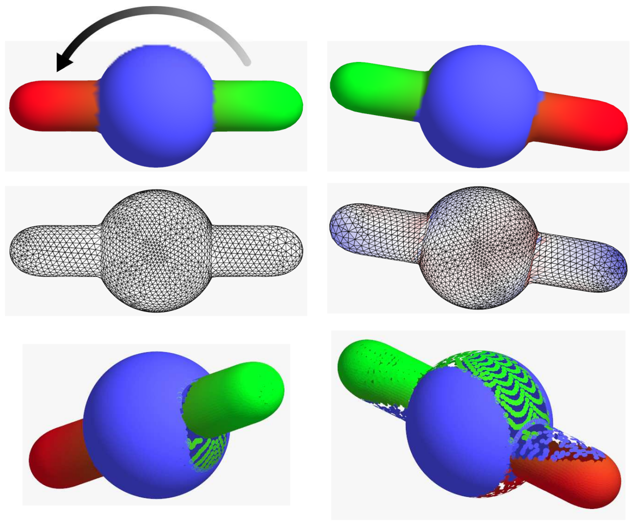
Fig. 10. A "stick" made from blended blobs (red to green) is rotated 180 degrees through a larger blue blob using Boolean union. Top row: tracking with colored mesh. Middle row: area distortion measure. Bottom row: points-only tracking.

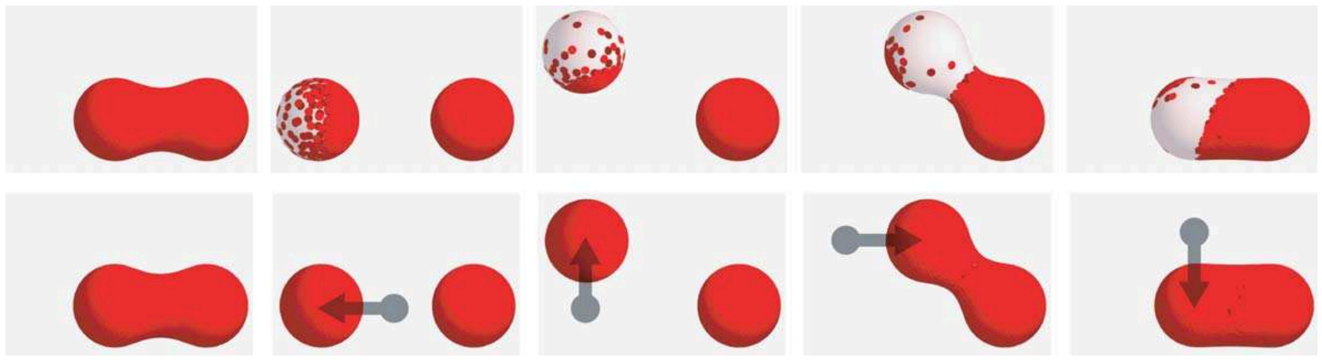

Fig. 8. Interactive manipulation of an implicit peanut, visualized using surfels that track the surface. The underlying implicit surface is also visualized. Top row: normal velocity. Bottom row: our total velocity approach.

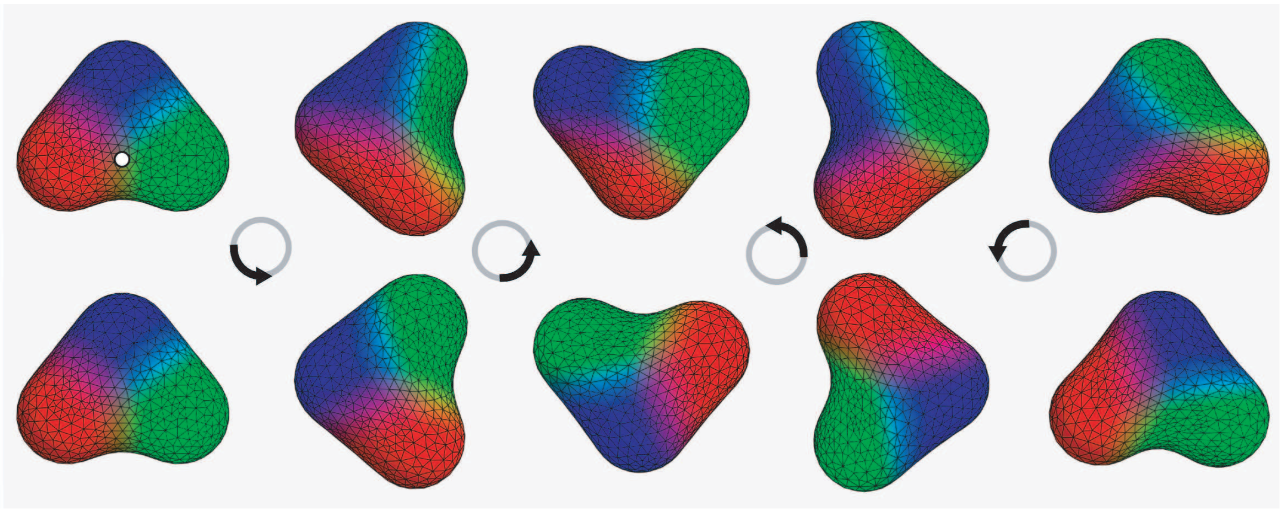

Fig. 9. 3 blended blobs rotating 360° about the origin (white dot) are tracked using a mesh generated at the first frame, with 10 fairing steps at each frame. Colors are assigned to vertices at the first frame, so the change in color shows how the vertices flow across the surface over time. With perfect tracking, the first and last frames would be identical. Top row: normal velocity. Bottom row: our total velocity approach.

6.2 Motion Blur

We have implemented our technique in the MAYA animation software that uses the Mental Ray renderer. Mental Ray requires the velocity of each point on a mesh in order to motion blur it. We first converted the blobby field into a mesh using some variant of the marching cubes algorithm [Lorensen and Kline 1987]. Then we used the results in Section 5 to compute a velocity for each vertex of the mesh.

Figure 11 shows a simple weighted average on the left and our motion blur on the right. Notice that our approach predicts a more correct velocity. Figure 12 shows a motion blurred sequence of an animation of a texture mapped lava-like flow.

7. CONCLUSION AND FUTURE WORK

In this paper we have presented a way to uniquely define a velocity at the surface of an evolving implicit function. We applied this method to two problems involving blobby implicit surfaces: tracking a moving surface and motion blurred rendering. Equations 1 and 4 are general and can be applied to the computation of the velocity field for any given implicit surface.

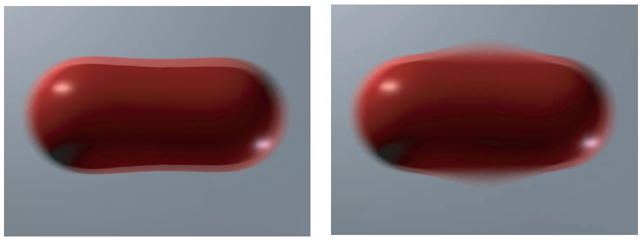

Fig. 11. Two motion blurred blobs moving towards each other. Left: simple weighting. Right: our approach.

We believe that our result could be useful in other applications of implicit surfaces, in particular the the problem of visualizing an implicit surface being interactively manipulated in a 3D modeling tool [Witkin and Heckbert 1994]. Figure 8 demonstrates that it is possible to maintain a reasonable surface sampling by advecting the particles through the velocity field defined by the user's actions.

As visualization is the bottleneck in interactive implicit modeling, this improvement could have significant impact. Another application area might be to use this result to improve particle based simulations of fluids.

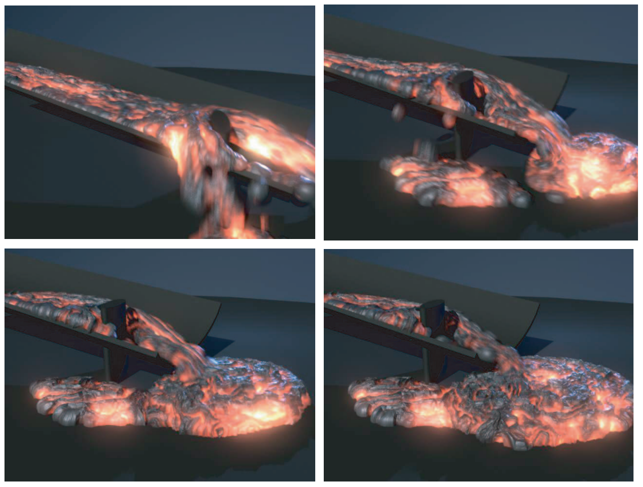

Fig. 12. A sequence of frames from a lava simulation.

ACKNOWLEDGMENTS

The authors thank Duncan Brinsmead and Igor Mordatch for helpful discussions. The second author was funded in part by NSERC and MITACS.

REFERENCES

BLINN, J. F. 1982. A Generalization of Algebraic Surface Drawing. *ACM Transactions on Graphics 1,* 3 (July), 235–256.

BOUTHORS, A. AND NESME, M. 2007. Twinned meshes for dynamic triangulation of implicit surfaces. In *Proc. Graphics Interface 2007*. 3–9.

BROCHU, T. AND BRIDSON, R. 2009. Robust topological operations for dynamic explicit surfaces. *SIAM Journal on Scientific Computing 31,* 4, 2472–2493.

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

LEVET, F., GRANIER, X., AND SCHLICK, C. 2007. Marchingparticles: Fast generation of particles for the sampling of implicit surfaces. *Computer Graphics & Geometry 9,* 1, 18–49.

LORENSEN, W. AND KLINE, H. 1987. Marching Cubes: A High Resolution 3D Surface Construction Algorithm. *ACM Computer Graphics (SIGGRAPH '87) 21,* 4 (August), 163–169.

MEYER, M., KIRBY, R. M., AND WHITAKER, R. 2007. Topology, accuracy, and quality of isosurface meshes using dynamic particles. *IEEE Transactions on Visualization and Computer Graphics 12,* 5.

MOORE, E. H. 1920. On the reciprocal of the general algebraic matrix. *Bulletin of the American Mathematical Society 26*, 394–395.

MULLAN, M., WHITAKER, R., AND HART, J. 2004. Procedural Level Sets. Presented at the NSF/DARPA CARGO meeting, May, 2004.

OHTAKE, Y., BELYAEV, A., AND PASKO, A. 2003. Dynamic mesh optimization for polygonized implicit surfaces with sharp features. *The Visual Computer 19,* 2-3, 115–126.

RODRIAN, H.-C. AND MOOCK, H. 1996. Dynamic triangulation of animated skeleton-based implicit surfaces. In *Proc. Implicit Surfaces '96*.

SMETS-SOLANES, J.-P. 1996. Vector Field Based Texture Mapping of Animated Implicit Objects. *Computer Graphics Forum (EUROGRAPHICS'96 Proceedings) 15,* 3, 289–300.

WITKIN, A. AND HECKBERT, P. 1994. Using Particles to Sample and Control Implicit Surfaces. *ACM Computer Graphics (SIGGRAPH '94) 28,* 4 (July), 227–234.

WYVILL, G., MCPHEETERS, C., AND WYVILL, B. 1986. Data structure for soft objects. *The Visual Computer 2,* 4, 227–234.