Tracking Surfaces with Evolving Topology

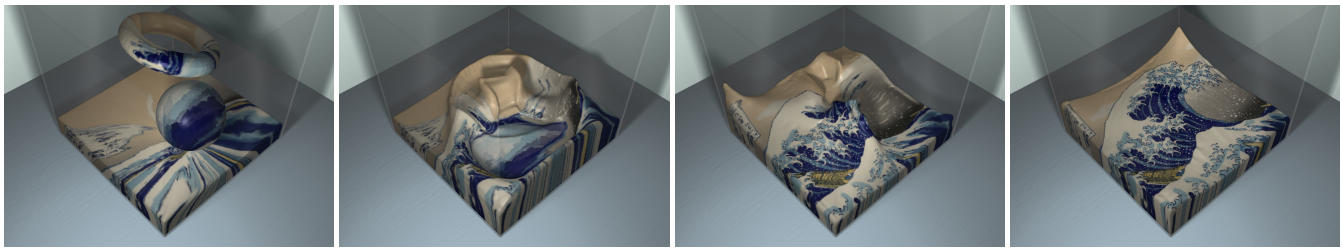

Figure 1: *Our method recovers a high-quality, temporally coherent triangle mesh from any sequence of closed surfaces with arbitrarily changing topology. We can reliably extract correspondences from a level set and track textures backwards through a fluid simulation.*

Abstract

We present a method for recovering a temporally coherent, deforming triangle mesh with arbitrarily changing topology from an incoherent sequence of static closed surfaces. We solve this problem using the surface geometry alone, without any prior information like surface templates or velocity fields. Our system combines a proven strategy for triangle mesh improvement, a robust multi-resolution non-rigid registration routine, and a reliable technique for changing surface mesh topology. We also introduce a novel topological constraint enforcement algorithm to ensure that the output and input always have the same topology. We apply our technique to a series of diverse input data from video reconstructions, physics simulations, and artistic morphs. The structured output of our algorithm allows us to efficiently track information like colors and displacement maps, recover velocity information, and solve PDEs on the mesh as a post process.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation;

Keywords: mesh deformation, non-rigid tracking, Implicit surfaces, fluid simulation

1 Introduction

Robust computational representations of deforming surfaces are considered indispensable within many scientific and industrial fields. Medical scientists deduce clues about the human body from the level sets of time-varying voxel data, physicists extract geometric information from simulations and acquisitions of fluid interfaces, and computer graphics professionals generate animations and capture performances in order to entertain audiences. As tools that *generate* time-evolving surfaces become increasingly commonplace, it is essential that we, as computer graphics researchers, provide better tools for the analysis and computational processing of these forms of animated geometry.

One particular class of evolving surface, namely surfaces that change topology through time, is particularly difficult to deal with. Because these surfaces are able to bend, split apart, reconnect themselves, and disappear through time, it is impossible to make any convenient assumptions about their shape and connectivity. For this reason, *implicit surfaces* such as contoured voxel data and metaballs, are extremely popular for representing such time-evolving surfaces. Unfortunately, these implicit surfaces are poorly suited for many important geometric tasks, such as mapping how surface points at one particular time correspond to surface points sometime later.

In this paper, we provide a general, robust method for tracking correspondence information through time for an arbitrary sequence of closed input surfaces. We do not require any context clues such as velocity information or shape priors, and we allow the surfaces to change topology through time. We solve this problem by combining a robust non-rigid registration algorithm, a reliable method for computing topology changes in triangle meshes, and a mesh-improvement routine for guaranteeing numerical accuracy and stability. The output of our method is a series of temporally coherent triangle meshes, as well as a *mesh event list* that tracks how surface vertices correspond through time.

We apply our method to data sets generated by different methods, such as physics simulations using two separate surfacing algorithms, morphing surfaces generated by implicit surfaces, and performance capture data reconstructed from videos. We show that we can reliably extract correspondence information that was absent from the original geometry, and we utilize this information to significantly enhance the input data. Using our algorithm, we are able to preserve important surface features, apply textures and displacement maps, simulate partial differential equations on the surface, and even propagate visual information *backwards* in time. When applied to dynamic shape reconstruction problems, we are able to reliably track the input without making any assumptions about how the data was generated. One can argue that this template-free tracking is an important tool for scientific experiments where it is essential to remove bias from the tools used for information discovery. The contributions of our work are as follows:

- We provide the first comprehensive framework for tracking time-varying closed surfaces where topology can change.
- Our algorithm is able to greatly enhance existing datasets with valuable temporal correspondence information. Some examples include displacement mapping of fluid simulations and texture mapping of level set morphs.
- We introduce a novel topology-aware wave simulation algorithm for enhancing the appearance of existing liquid simulations while significantly reducing the noise present in similar approaches.
- Because our method robustly extracts surface information from input data alone, we provide a reliable way to automatically track markerless performance capture data without the need for a template.

2 Related Work

Our work is closest to a recent publication of Stam and Schmidt [2011]. They showed that, by examining the input parameters for an implicit surface algorithm, one can derive the surface velocity to create motion blur and more coherent surface animations. By integrating surface velocity through time, they presented a method to approximate point-to-point correspondences which can be used to track texture information. This inspirational work introduced some exciting applications for tracking correspondences through complicated deformations, and we believe that it brought the community a significant step closer to solving the general problem of tracking a topology-evolving surface. Our method is different from theirs in a number of ways. Firstly, we wish to solve the more general problem of tracking an arbitrary input surface sequence, so we do not assume that we know the parameters behind the surface dynamics. Secondly, their correspondence information is only as accurate as their velocity integration, so it is prone to numerical drift (especially with the first order integrators commonly used in graphics applications — more accurate Runge-Kutta integrators would pose the additional constraint that we have the ability to evaluate the input parameters with random access in time). Our method uses a nonlinear shape matching optimization to minimize this drift, and the difference is especially apparent in the presence of large rotations.

To the best of our knowledge, our method is the first to provide a solution to the problem of registration combined with topology change. For the remainder of this section, we divide the works most related to ours into two camps: those related to deformable shape matching and registration, and those related to surface evolution with topology changes.

Deformable Shape Matching and Registration The field of dynamic geometry processing is actively involved in the problem of automatic correspondence extraction from inconsistent dynamic meshes [Chang et al. 2010]. Dense, inter-frame surface correspondences describe a space curve for each surface sample and provide rich information for temporal mesh analysis and surface tracking—hence their importance in modern data capture problems such as marker-free human performance tracking and dynamic shape reconstruction from incomplete 3D scan sequences. Time-varying data processing typically involves special treatment and consideration of temporal coherency. We will therefore focus our discussion on methods that take sequences of meshes or point clouds as input.

Most methods that establish full surface correspondences through time rely on an existing template model or construct it in a separate step. With a *fixed topology* and known geometric state, template models are popular since they significantly simplify the reconstruction problem of geometry and motion. Mitra et al. [2007] introduced a registration method that aggregates scan sequences into a 4D space-time surface to build a more complete template. While inter-frame correspondences can be estimated from kinematic surface properties, this technique is limited to fairly small deformations in the input data and does not allow scans to change topologies. To handle more complex deformations, the work of [Süßmuth et al. 2008] tracks a pre-extracted template shape using a rigidity maximizing deformation model, but is still sensitive to topology-varying input data. The statistical framework introduced by Wand et al. [2007] and later improved in [Wand et al. 2009] estimates a globally consistent template model with a fix topology from real-time acquisitions of input point clouds. While being restricted to slowly-varying surface deformations, their methods can identify topology variations in the input data. The geometry and motion reconstruction technique described in [Li et al. 2009] uses a rough geometric approximation of a pre-constructed template model to prevent wrong topology estimations during dynamic shape tracking. Although the topology of the reconstruction is static, this purely geometry based technique includes a particularly robust non-rigid registration algorithm that can handle significantly larger deformations than previous methods. While highly disruptive motions are explicitly treated in the global framework of Tevs et al. [2012], highly incomplete acquisition data can still damage the template extraction. Due to the difficulty of constructing a consistent template model from incomplete data, several recent research have focussed in introducing structural priors such as skeletons [Zheng et al. 2010] or explicit parameters for joint positions and skinning weights [Chang and Zwicker 2011]. These methods are particularly reliable for articulated subjects but are not suitable for scenarios such as clothed human performances.

While correspondences are desirable for many geometric analysis and manipulation purposes, several state-of-the-art shape completion methods skip this requirement and do not extract a template model in order to achieve a drift-free treatment of topology varying input data. The technique presented in [Sharf et al. 2008] is able to produce a watertight surface sequence from extremely noisy input scans using a volumetric incompressible flow prior but suffers from significant flickering in the reconstruction. In the context of fluid capture, Wang et al. [2009] demonstrated a framework to fill in holes in partially captured liquid surfaces using a physically guided model. Their method achieves time-coherent reconstructions of dynamic surfaces but are restricted to fluid simulations since frame-to-frame correspondences are guided by the velocity of a fluid simulation. Lately, Li et al. [2012] demonstrated a shape completion framework for temporally coherent hole filling of incomplete and flickering affected scans of human performances. Their method makes minimal assumptions about the surface deformation by establishing correspondences within a small time window but side steps the extraction of globally consistent correspondences through time.

The proposed technique is able to establish full correspondences across time-series of input meshes and is not limited to a fixed topology like template-based methods. Our method is grounded on a general purpose non-rigid registration algorithm similar to [Li et al. 2009; Li et al. 2012] and can therefore be applied widely, ranging from fluid surface dynamics, human body performances, and arbitrary shape morphings.

Surface Evolution with Topology Changes Several methods exist for tracking topology-changing surfaces through time with the aid of prescribed motions or velocity fields. Level set methods [Osher and Fedkiw 2003] and particle level set methods [Enright et al. 2002] are a popular method for representing a dynamic implicit surface. These methods consider the zero level set of a voxelized signed distance function, and they integrate velocity information in order to move the function. This integration displaces the zero set of the function, resulting in a moving surface. Müller [2009] used a strategy of repeatedly re-sampling an evolving Lagrangian triangle mesh in order to provide fast surface tracking for fluid surfaces. Semi-Lagrangian contouring [Bargteil et al. 2006a] also utilizes Lagrangian information in the form of extracted surface geometry in order to improve surface tracking. Bargteil et al [2006b] and Kwatra et al. [2007] illustrate the ability to track texture information in a fluid simulation, enhance their animations by synthesizing new texture as the surface evolves. Dinh et al. [Dinh et al. 2005] also tracks texture information on a topology-changing implicit surface by solving an optimization problem over space and time.

The surface evolvers most similar to ours are mesh-based surface tracking methods [Du et al. 2006; Wojtan et al. 2010; Brochu et al. 2010]. The idea behind these techniques is to evolve a triangle mesh

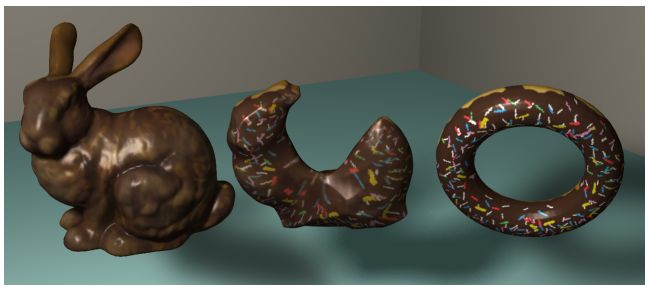

Figure 2: *A morphing example where surface textures are tracked. Unlike existing techniques, our method does not exhibit ghosting artifacts.*

according to a velocity field, which allows for better preservation of geometric features and correspondence information than using an implicit surface. These mesh-based methods go hand-in-hand with robust numerical methods for changing mesh topology [Brochu and Bridson 2009; Wojtan et al. 2009; Campen and Kobbelt 2010; Zaharescu et al. 2007; Pons and Boissonnat 2007]. Within our framework, we use a method similar to Wojtan et al. [2009] for changing mesh topology, because of its speed and versatility (Further details are explained in §4.3).

While each of these works on surface evolution certainly helped inspire ours, we would like to remind the reader that our method solves a significantly different problem of tracking *without any velocity information*. In this light, we do not perceive our method as a competitor to existing fluid simulation techniques, but as a powerful enhancement tool — it allows a user to convert the output from *any simulation type* into a temporally coherent deforming mesh sequence. Our tracked surfaces are a great improvement over implicit surfaces in the information they provide, the details they preserve, and the useful applications that they aid.

3 Problem Statement

This paper is concerned with the problem of taking a series of closed surfaces through time as input, and then replacing these surfaces with a single, temporally coherent deforming triangle mesh. We wish to allow these input surfaces to have arbitrary shapes and topology, and these shapes and topology are allowed to change significantly from one surface to the next. Because such data can come from a range of diverse sources in practice, we cannot assume any specific domain knowledge, nor can we assume that we are given additional information such as velocity fields. While surface tracking and registration is a widely studied problem, we are unaware of any tracking methods that are both robust to large deformations and arbitrarily complicated topology changes while retaining correspondence information. This is unfortunate, because frequent topology changes result from many common sources such as fluid dynamics, morphing, and erroneous scanned data.

To adequately solve this problem, we must define what it means for two shapes to correspond in the presence of topology changes and find the most appropriate mapping between consecutive pairs of input surfaces. This correspondence information should gracefully propagate through changes in surface topology. We require our method to handle arbitrarily large plastic deformations through time while keeping the computation tractable.

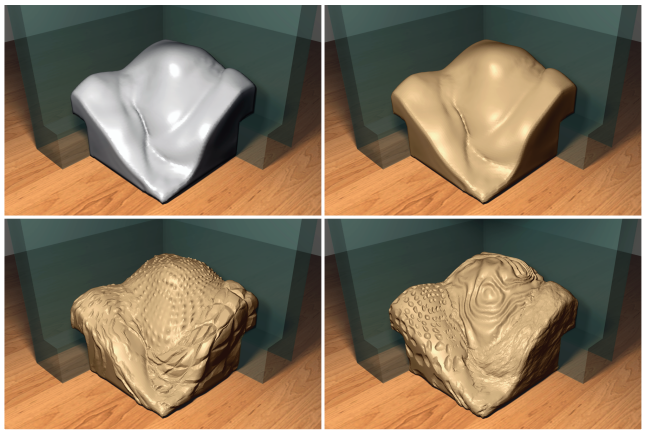

Figure 3: *Our method can turn a temporally incoherent mesh (upper left) into a coherent one (upper right). We use this tracked mesh to add displacement maps as a post-process without having to resimulate any physics.*

4 Method

Our algorithm consists of several interwoven operations: mesh improvement (§4.1), non-rigid alignment (§4.2), and topological change (§4.3). The mesh improvement operation ensures that our output mesh retains high quality triangles while only minimally resampling geometry. The non-rigid alignment step ensures that our output mesh actually conforms to the desired shapes through time, and the topology change step ensures that the topology of our output mesh conforms to that of the desired input shapes in each frame. We show that these three operations alone are enough to generate a smoothly deforming mesh with high quality geometry. However, in order to utilize these deforming meshes to their full extent, we also record correspondence information along the way (§4.4). Finally, we explain how to use the recorded correspondence information to efficiently propagate information forward and backwards through time as a post-process (§4.6).

4.1 Mesh Improvement

A detailed surface mesh with well-shaped triangles is essential for a wide variety of beneficial computations. In addition to enhancing numerical stability in our non-rigid registration solver (§4.2) as well as the geometric intersection code in our topology change routine (§4.3), a triangle mesh free from degeneracies is necessary for such basic operations as interpolation, ray tracing, and collision detection. As we explain later in §5, the guaranteed mesh quality from our algorithm allows us to densely sample complex textures, generate displacement maps which are less prone to self-intersections, and solve partial differential equations on a deforming mesh using a finite element method.

In our framework, we follow the mesh improvement procedures outlined by Wojtan et al. [2011]. When edges become too long, we split them in half by adding a new vertex at the midpoint. When edges become too short, or when triangle interior angles or dihedral angles become too small, we perform an edge collapse by replacing an edge with a single vertex. Although we did not implement them for this project, edge flips are also another excellent mesh resampling operation.

When improving a dynamically-deforming mesh, the main challenge is to find the right balance between high quality triangles

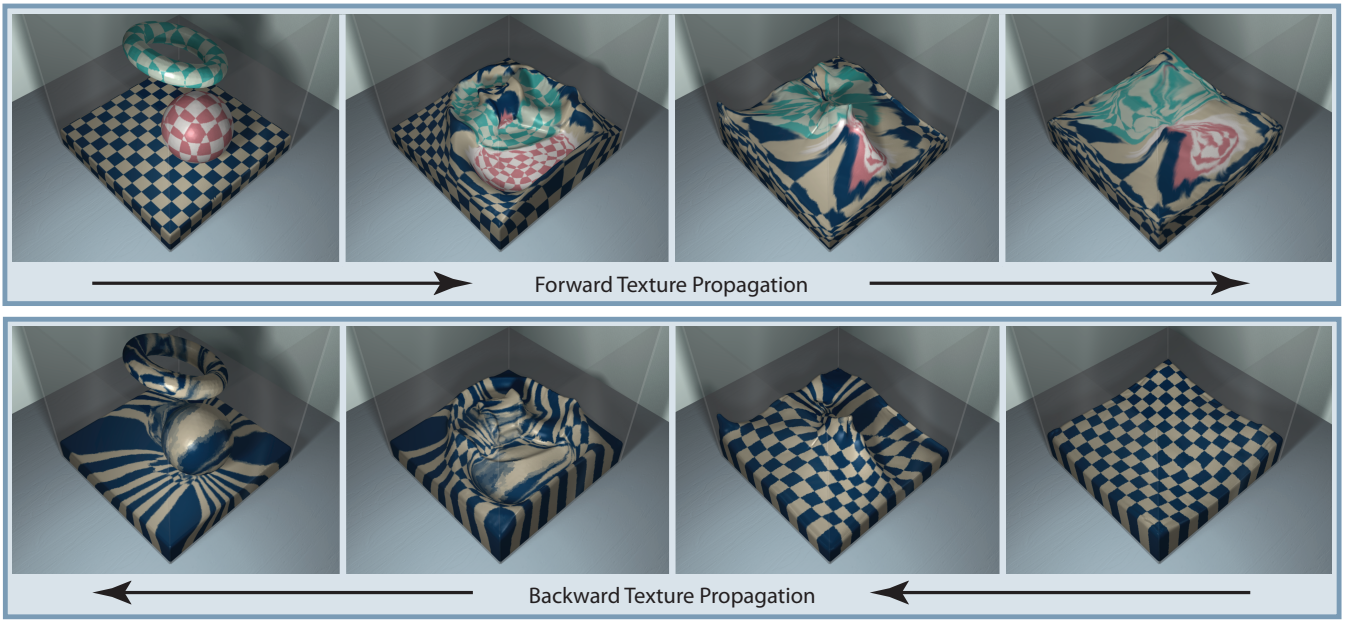

Figure 4: *These animations show how we can use our algorithm to propagate a texture both forwards and backwards through time. In the bottom animation, the fluid simulation naturally splashes around as it settles into on a checker texture.*

and excessive vertex re-sampling. Though we are free to customize these mesh improvement parameters however we like, we used similar parameters for all of the examples in this paper. We used a minimum interior angle of 10 degrees, a minimum dihedral angle of 45 degrees, and a maximum:minimum edge length ratio of 4:1. For a more in depth discussion on choosing parameters for these operations, please see [Wojtan et al. 2011].

4.2 Non-Rigid Registration

In a continuous space-time setting, it is impossible to establish one-to-one correspondences (i.e., creating a diffeomorphism) between consecutive meshes with unrelated topology. However, to construct a natural mapping between discrete surfaces that smoothly deform and change topology, we define correspondences between consecutive frames as those that warp the source shape onto the target while minimizing surface distance and shape distortion (scaling and stretching). The rationale behind this classic non-rigid registration approach is to maximize geometric shape compatibility since it is the only available information. To account for the large resolutions and deformation complexity of our examples, we adapted the state-of-the-art bi-resolution registration algorithm introduced in [Li et al. 2009] to compute accurate correspondences efficiently. While the original technique is designed to handle pairs of partial scans where only a sub-region contains correspondences (part-in-part matching), our setting is easier in that two complete shapes are available (whole-in-whole matching). Hence, assuming that every source point has a target allows us to improve convergence of the two-stage optimization with minimal changes as illustrated next:

Coarse-level Non-Linear Optimization. We first estimate correspondences between two shapes by bringing them into alignment using a sampled warp field that maximizes local rigidity. As shown in [Li et al. 2009], this non-linear problem can be effectively solved using a non-rigid *iterative closest point* (ICP) algorithm. The idea is to iterate between closest point estimations and surface deformation where rigidity is gradually decreased whenever convergence of the optimization is detected. Instead of computing closest points as in previous work, our method uses correspondence estimates projected in the surface normal directions. Since ideally we wish to create a bijective mapping between two complete shapes, we found that this heuristic significantly reduces target surface regions that do not correspond with the source. We only prune correspondences that have surface normals that are more than 60 degrees apart.

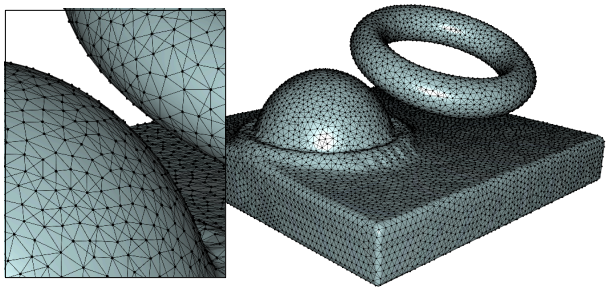

Figure 5: *Our mesh is augmented with deformation graph for a robust coarse-level non-rigid registration. We use geodesic distances to construct the graph to avoid edge connections between disconnected components.*

Similar to the rigidity-maximizing deformation model described in [Li et al. 2009], we augment the high-resolution mesh with a lower complexity *deformation graph* and solve for the affine transformation $(A_i, \mathbf{b}_i)$ of every graph node instead of the vertices. The affine transformations are then transferred to the vertices via linear blend skinning. In particular, we describe the motion of each vertex by a linear combination of the transformations of the $k = 4$ nearest nodes, weighted by the inverse of the geodesic distance to each node. This choice of using geodesic distances is important for two

reasons: Firstly, geodesic distances avoid graph edge connections between disconnected but nearby surfaces. Secondly, they allow vertex motions to ignore graph nodes that are close in Euclidean distance but far in geodesic distance. Because we wish to handle arbitrary deformations like fluids, we simply use a high-resolution uniformly sampled graph where the distance between nodes is 4 times larger than the average edge length of the underlying mesh.

Once the deformation graph is constructed and the mesh vertices weighted, we solve for the optimal transformation of the graph nodes that minimizes a fitting energy combining a point-to-plane and point-to-point distance for each mesh vertex: $E_{\text{fit}} = E_{\text{plane}} + 0.1\ E_{\text{point}}$. To achieve a smooth as-rigid-as-possible regularization in the deformation, we minimize the deviation of A_i from a true rotation and only allow nodes to affect their edge-connected neighbors. These two components form the regularization term $E_{\text{reg}} = E_{\text{rigid}} + 0.1E_{\text{smooth}}$. The total energy $E_{\text{tot}} = \alpha_{\text{fit}}E_{\text{fit}} + \alpha_{\text{reg}}E_{\text{reg}}$ is solved using a standard Gauss-Newton solver based on Cholesky decomposition. We alternate between correspondence point estimation and surface deformation until convergence and gradually relax the regularization by dividing α_{reg} by 10. For each pair of consecutive frames we initialize the optimization with $\alpha_{\text{fit}} = 0.1$ and $\alpha_{\text{reg}} = 1000$.

Fine-Scale Linear Optimization. While multiple iterations of the coarse level optimization make sure that large deformations between source and target shape are recovered, a second warping step uses a more efficient (but rotation sensitive) linear mesh deformation technique to capture the full geometric details of the raw input meshes. The optimization uses the same fitting term E_{fit} as described before and solves for the displacement of each vertex by minimizing the difference between adjacent vertex displacements and the variation in edge lengths using $E_{\text{reg}} = E_{\text{disp}} + E_{\text{edge}}$. To avoid self intersections, we prune correspondences that are further than a threshold $\sigma = 0.1$. Finally, we synthesize fine-scale details from the target on the pre-aligned mesh by minimizing $E_{\text{tot}} = E_{\text{fit}} + E_{\text{reg}}$ using an efficient conjugate gradient solver. Despite the robustness of the proposed non-rigid registration approach, we do not guarantee that every target surface region will have a corresponding source point. Such cases require a change in topology.

4.3 Topological Change

This paper considers a more general class of input deformations than most previous methods — we aim to track surfaces that are not only highly deformable, but that may change topology arbitrarily through time. For example, we allow new surface components to appear from nowhere in the middle of an animation, and we expect that entirely disparate surface regions may suddenly merge together. In order to accurately track such extreme behavior in the input data, we build new tools to constrain the topology of our mesh to that of an arbitrary closed input surface.

We base our topology change method on that of Wojtan et al. [2009] with subdivision stitching [2010] as explained in their SIGGRAPH course [2011]. This method begins by comparing a mesh $\mathcal{M}$ to its voxelized signed distance function $\Phi_{SDF}(\mathcal{M})$. Then, wherever there are local topological differences between $\mathcal{M}$ and $\Phi_{SDF}(\mathcal{M})$, the method replaces triangles from the input mesh with triangles from the extracted isosurface of $\Phi_{SDF}(\mathcal{M})$. This strategy effectively forces the explicit triangle mesh to change such that its topology matches that of its voxelized signed distance function.

We chose to use this method primarily because of its flexibility and robustness. We would like the surface to change topology not only when the mesh intersects itself, but also whenever the input geometry happens to change its own topology. Furthermore, because this method is independent of surface velocity, it adds another layer of robustness to our algorithm; in the event that our registration routine produces inaccurate displacement information, the topology algorithm will correct the final shape by drawing new surface geometry directly from the input.

To do this, we generalize the idea of Wojtan et al.; instead of constraining the topology of the input mesh to match that of its own signed distance function, we constrain the input mesh to match the topology of *any voxelized implicit surface*. We simply voxelize an arbitrary implicit surface Θ, and replace the signed distance function $\Phi_{SDF}(\mathcal{M})$ in the original with our new function Θ. The algorithm then compares the topology of the mesh $\mathcal{M}$ to the topology of Θ, and replaces $\mathcal{M}$'s triangles with triangles from the extracted isosurface of Θ wherever $\mathcal{M}$ and Θ have a different local topology. We can refer to this generalized topology change routine as ConstrainTopology$(\mathcal{M}, \Theta)$. Using this terminology, the original algorithm of Wojtan et al. can be executed by calling ConstrainTopology$(\mathcal{M}, \Phi_{SDF}(\mathcal{M}))$.

Within our deformation framework, we use this generalized topology change algorithm in two ways: first to ensure that the deforming mesh changes topology if it intersects itself, and second, to ensure that the deforming mesh has the same topology as the target input data. These actions can be computed by calling ConstrainTopology$(\mathcal{M}, \Phi_{SDF}(\mathcal{M}))$ and ConstrainTopology$(\mathcal{M}, \Phi_{SDF}(\mathcal{T}))$, respectively, where $\mathcal{T}$ is the target mesh from the input data. We will specify the exact order in which to call these functions in section §4.5.

4.4 Recording Correspondence Information

Throughout the computation of our deforming mesh $\mathcal{M}$, we want to track how its correspondences evolve through time. The previously mentioned mesh modification routines can cause significant changes in correspondence information, and we must track how these changes occur.

The mesh deformation algorithm described in §4.2 is Lagrangian in nature, so it moves individual vertices to their new locations at each frame in the animation sequence. Consequently, the vast majority of vertex locations in our mesh at a given frame number correspond exactly to the location of that same vertex at earlier and later frames numbers. For these vertices, information about their corresponding position at different points in the sequence is implicit; vertex $\mathbf{v}_i$ at frame number j corresponds exactly with $\mathbf{v}_i$ at frame $j + 1$.

The only vertices which do *not* have this trivial correspondence with vertices in different frames are the few vertices which were created or destroyed due to re-sampling. Within our framework, the only way to create new vertices is via topological change (§4.3) or edge and triangle subdivision (§4.1). The only way for us to destroy vertices is via topological change (§4.3) or edge collapse (§4.1). Note that some other potential mesh improvement procedures like mesh fairing [Jiao 2007; Brochu and Bridson 2009; Stam and Schmidt 2011] improve triangle quality at the expense of re-sampling correspondence information by diffusing it along the surface. For this reason, we did not use such fairing procedures in §4.1.

For each transition between two frames, we track these re-sampling events (edge subdivision, triangle subdivision, edge collapse, topology change) in what we call an **event list**. The event list stores detailed information about each re-sampling event, and it is sorted by the order in which the re-sampling events took place. Each event in the list records information of the form $(V_{\text{in}}, V_{\text{out}}, f(V_{\text{in}}), g(V_{\text{out}}))$, where V_{in} is a set of the input vertices, V_{out} is a set of the output vertices, $f(V_{\text{in}})$ is a function that assigns information to V_{out} as a function of V_{in}, in case we want

to propagate information forwards. Similarly, $g(V_{\text{out}})$ is a function that assigns information to V_{in} as a function of V_{out}, in case we want to propagate information backwards.

When we subdivide an edge in the mesh, a vertex $\mathbf{v}_{\text{new}}$ is created at a location somewhere in between two endpoints $\mathbf{v}_A$ and $\mathbf{v}_B$. The event list records $(\{\mathbf{v}_A, \mathbf{v}_B\}, \{\mathbf{v}_A, \mathbf{v}_B\, \mathbf{v}_{\text{new}}\}, \mathbf{v}_{\text{new}} := (\mathbf{v}_A + \mathbf{v}_B)/2, \text{null})$. For a triangle subdivision event, we insert a new vertex $\mathbf{v}_{\text{new}}$ inside the triangle and split the triangle into three pieces. The event list records: $(\{\mathbf{v}_A, \mathbf{v}_B, \mathbf{v}_C\}, \{\mathbf{v}_A, \mathbf{v}_B, \mathbf{v}_C, \mathbf{v}_{\text{new}}\}, \mathbf{v}_{\text{new}} := \alpha\mathbf{v}_A + \beta\mathbf{v}_2 + \gamma\mathbf{v}_3, \text{null})$, where α, β, and γ are barycentric coordinates. When we collapse an edge in our mesh with endpoints $\mathbf{v}_A$ and $\mathbf{v}_B$, we remove the edge endpoints from the mesh and place a new vertex $\mathbf{v}_{\text{new}}$ at the center of the collapsed edge. The event list records: $(\{\mathbf{v}_A, \mathbf{v}_B\}, \mathbf{v}_{\text{new}}, \mathbf{v}_{\text{new}} := (\mathbf{v}_A + \mathbf{v}_B)/2, \mathbf{v}_A := \mathbf{v}_B := \mathbf{v}_{\text{new}}\}$.

When a topological change occurs, surfaces can split wide open and entire patches of new geometry can be created. For each patch of new geometry after the topology change, we propagate information from the vertices on the boundary of the patch inward, using a breadth-first graph marching algorithm (similar to Yu et al. [2012]). Though several propagation strategies are valid at this point (during the marching algorithm, each new vertex could simply copy information from its nearest neighbor, it could distribute information evenly throughout the patch, e.g. by solving an elliptic PDE, etc.), we chose a strategy of each vertex taking the average of the information from its visited neighbors during the breadth-first march. For each new vertex that is created, our event list records the list of boundary vertices, the new vertex, and the linear combination of boundary vertices that results from this marching and averaging. There is no backward correspondence assignment for these vertices.

Lastly, vertices can be deleted in a topological merge. We treat such operations the same way that we treat new vertices that result from a topology change, but in reverse: before the patch of vertices is destroyed, we march inward from the boundary of the patch of deleted vertices and propagate information using the same averaged vertex scheme. For each vertex that is deleted, our event list records the list of boundary vertices, the new vertex, a null forward operation, and the linear combination of boundary vertices that results from the marching and averaging operation.

4.5 Summary of the Tracking Algorithm

We review the steps of our tracking method in Algorithm 1. Our method begins by initializing a triangle mesh $\mathcal{M}$ to the first frame $\mathcal{T}_0$ of a mesh sequence. We then immediately call our mesh improvement routine (§4.1), which ensures that $\mathcal{M}$ consists of high quality geometry. Next, we enter the main loop of our algorithm, which visits each of the input meshes $\mathcal{T}_j$ in turn. For each input mesh, we use our course non-rigid registration routine (§4.2) to align the low-resolution features of $\mathcal{M}$ as closely as possible with those of $\mathcal{T}_j$. Once this coarse alignment has terminated, we perform a fine-scale linearized registration in order to ensure that all of the high-resolution details of $\mathcal{M}$ line up with $\mathcal{T}_j$. At this point in the algorithm, we have deformed our mesh $\mathcal{M}$ such that it lines up with the input data frame T_j. This deformation may cause the triangles of $\mathcal{M}$ to stretch and compress arbitrarily, so we again perform a mesh improvement in order to clean up the overly deformed elements of $\mathcal{M}$.

Next, we must account for the fact that our mesh $\mathcal{M}$ may have changed topology. We execute the basic topology change algorithm in §4.3 by first computing a voxelized signed distance function near the surface of $\mathcal{M}$ and then ensuring that $\mathcal{M}$ has the same topology as the zero isosurface of this function. This step mainly cleans up any large self-intersections in the mesh by merging surface patches

1: Mesh $\mathcal{M}$ = LoadTargetMesh($\mathcal{T}_0$)
2: ImproveMesh($\mathcal{M}$)
3: **for** frame $j = 1 \rightarrow n$ **do**
4: LoadTargetMesh($\mathcal{T}_j$)
5: CoarseNonRigidRegistration($\mathcal{M}$, $\mathcal{T}_j$)
6: FineLinearRegistration($\mathcal{M}$, $\mathcal{T}_j$)
7: ImproveMesh($\mathcal{M}$)
8: $\Phi_{\text{SDF}}(\mathcal{M})$ = CalculateSignedDistance($\mathcal{M}$)
9: ConstrainTopology($\mathcal{M}$, $\Phi_{\text{SDF}}(\mathcal{M})$)
10: $\Phi_{\text{SDF}}(\mathcal{T}_j)$ = CalculateSignedDistance($\mathcal{T}_j$)
11: ConstrainTopology($\mathcal{M}$, $\Phi_{\text{SDF}}(\mathcal{T}_j)$)
12: ImproveMesh($\mathcal{M}$)
13: SaveEventListToDisk(j)
14: SaveMeshToDisk($\mathcal{M}$)
15: **end for**

algorithm 1: *Pseudocode for our mesh-tracking algorithm.*

together. Next, we execute a topology change algorithm again, but this time we constrain $\mathcal{M}$ to match the topology of the *input* mesh $\mathcal{T}_j$. This step ensures that we split apart any surfaces in $\mathcal{M}$ which stretches over gaps in T_j, as well as merging any separate regions of $\mathcal{M}$ that are actually merged in the input data. This extra topology constraint also acts as a fail-safe by re-sampling parts of $\mathcal{M}$ in the rare event that the registration algorithm was unable to find good matches between $\mathcal{M}$ and T_j.

At this point in the algorithm, our mesh $\mathcal{M}$ can consist of triangles with arbitrarily poor aspect ratios, because the topological sewing algorithm only cares about the connectivity of the mesh and not the condition of the individual mesh elements. We therefore call our mesh improvement routine once again to ensure that the mesh is fit for another round of tracking the input data. Note that throughout this entire algorithm, we document any re-sampling operations that occur (potentially in lines 2, 7, 9, 11, and 12 of Algorithm 1) and add them to our event list (§4.4). In the final two steps of this loop, we save our event list and the mesh $\mathcal{M}$ itself to disk. We then start the loop again with the next frame of animation T_{j+1}.

4.6 Propagating Information as a Post-Process

After we have finished tracking the input geometry (after all of the steps in §4.5 have run until completion), we have a series of temporally coherent animation frames of a mesh $\mathcal{M}$ that deforms and changes topology. Furthermore, we also have a per-frame *event list* that describes exactly how correspondences propagate throughout the animation. We can use this list to pass information like surface texture and surface velocity from one frame to the next. To pass information forward in time, we run through the event list in the order that each event took place, and, using the notation from section 4.4, we pass information to re-sampled vertices using the function $f(V_{\text{in}})$. Similarly, we pass information backwards in time by running backwards through the event list and using $g(V_{\text{out}})$.

5 Applications

Having detailed our method for obtaining a temporally coherent parameterization of an arbitrary sequence of closed manifold meshes (§4), we shift our focus to applications. We show how we can apply our method to track a broad range of different incoherent surfaces and how we can exploit extracted correspondence information to significantly enhance the meshes in variety of different ways.

Displacement Maps. Our first example shows three viscoelastic balls dropping on top of each other (Figure 3) generated

by a physically-based Eulerian simulator with a periodically re-sampling surface tracker similar to [Müller 2009]. Our method faithfully conforms to the target shape in every frame with minimal re-sampling.

We showcase our temporally coherent parameterization, free of noticeable drift, and high mesh quality, by applying two different sets of displacement maps to the simulation as a post-process. A displacement map consist of a per-vertex scalar, designating the amount to displace the vertex in the normal direction. We use our data structure (§4.6) to propagate displacements applied in the first frame to all later frames. Compared to tracking the simulation, this is almost instantaneous, taking only takes a few seconds for the entire animation. Swapping in different a different displacement map is fast and effortless. Compare this to the state of the art without our method, where an animator instead would have to re-simulate everything to change the geometry.

Color. Our second example shows a splashy liquid scene (Figure 4). This animation comes from a standard Eulerian solver using the Level Set Method [Osher and Fedkiw 2003] to track the free-surface. We track an incoherent sequence of marching cubes reconstructions of level sets from the simulation.

Similar to displacement maps, we may propagate colors applied in the first frame to all later frames. Our accompanying video shows a checkerboard pattern and a lava texture propagated through time. Further exploiting our temporal data structure, we also propagate a colors applied in the last frame *backwards* in time to the first frame (Figure 4). This technique allows us to enhance the splashy animation with an interesting artistic expression where an image is slowly revealed as the dynamics settle (Figure 1).

Wave simulation. Our third example improves the fidelity of the level set simulation by adding an extra layer of dynamics as a post-process (Figure 7). Because our method yields particularly high quality surface triangles with minimal re-sampling, we are able to use the resulting mesh to solve partial differential equations. Inspired by recent fluid animation research [Thürey et al. 2010; Yu et al. 2012], we augment our surfaces with a time-varying displacement map, computed as the solution to a second order wave equation:

$$\frac{\partial^2 h}{\partial t^2} = c^2 \nabla^2 h. \tag{1}$$

Here, h is wave displacement in the normal direction, ∇^2 is the discrete Laplace operator computed with cotangent weights [Botsch et al. 2010], and c is a user-chosen wave speed. We use our transition graph to transfer the state variables (wave heights h and velocities in the normal direction v) from one frame to the next, and we integrate the system using symplectic Euler integration with several sub-cycled time steps per input frame. One may optionally choose to add artificial damping to the simulation for artistic reasons by multiplying h by a $(1 - \epsilon)$ factor each step. No artificial damping was used in our simulations.

Our wave simulation method is novel in that it retains tight control over wave energy sources. We only add wave heights precisely at the locations in space-time where topological changes occur. This stands in opposition to previous work, which recomputes wave heights every time step based on surface geometry. The result of this distinction is that our simulations are much less likely to introduce energy due to numerical errors. Our simulations have a dramatically high signal-to-noise ratio – we can clearly see interesting wave interference patterns persist throughout the entire simulation.

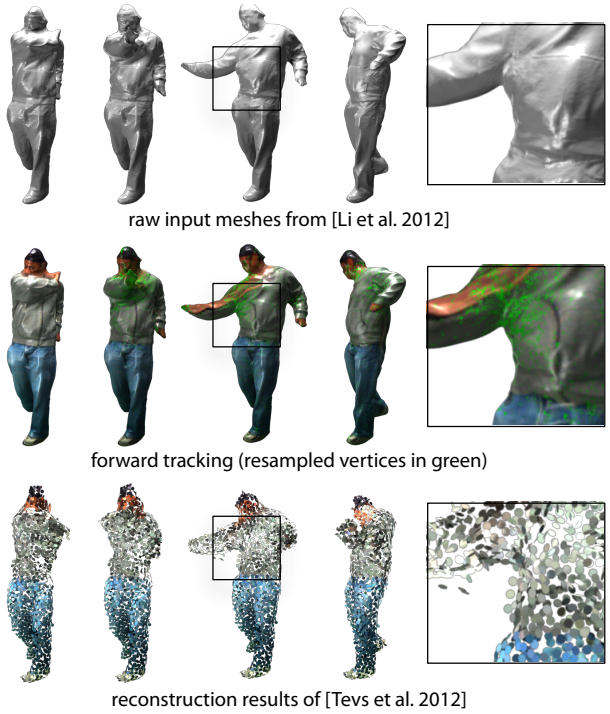

Figure 6: *Top: While being a closed manifold, our input performance capture data does have consistent vertices across frames and exhibits difficult topological variations. Middle: our method seamlessly handles topology changes and ensures high quality triangles. Resampled vertices from our mesh improvement algorithm are marked in green. Bottom: In addition to being expensive, the state-of-the-art animation reconstruction method of Tevs et al. fails at capturing the correct motion.*

Morph. Another application of our method is transferring colors through morphs that change topology between arbitrary genera (Figure 2). We use a simple linear blend between signed distance functions to create the morph and subsequently obtain a coherent mesh by tracking it with our framework. We start by propagating color backwards from the final frame propagation, and then we use the colors which were propagated to the first frame to obtain a base texture. In this way an artist can see where important feature points end up on the target shape to aid in creating a more natural morph. To obtain a really high quality morph we may additionally blend between the two forward and backward propagated colors.

Performance Capture. One final application of our method is in performance capture. Unlike previous methods, we are able track captured data that has topology changes due to occlusion while obtaining temporally coherent correspondences (Figure 6). We apply a texture in the first frame and propagate it forward. Regions that are unoccluded throughout the sequence are tracked faithfully.

6 Evaluation

We performed an extensive series of tests to evaluate our method. We used the viscoelastic simulation input (Figure 3) as a testbed while we varied parameters, turned off various parts of our code, and attempted alternative approaches. Please see our accompanying video for visualizations of these tests.

In Figure 9, we show how our method compares to the naïve ap-

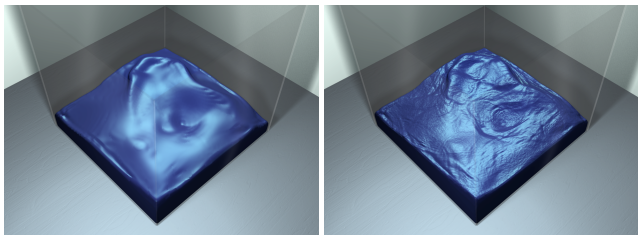

Figure 7: *Our framework allows us to synthesize high-frequency details of a separate wave simulations (right) on top of a lower resolution pre-simulated fluid surface (left).*

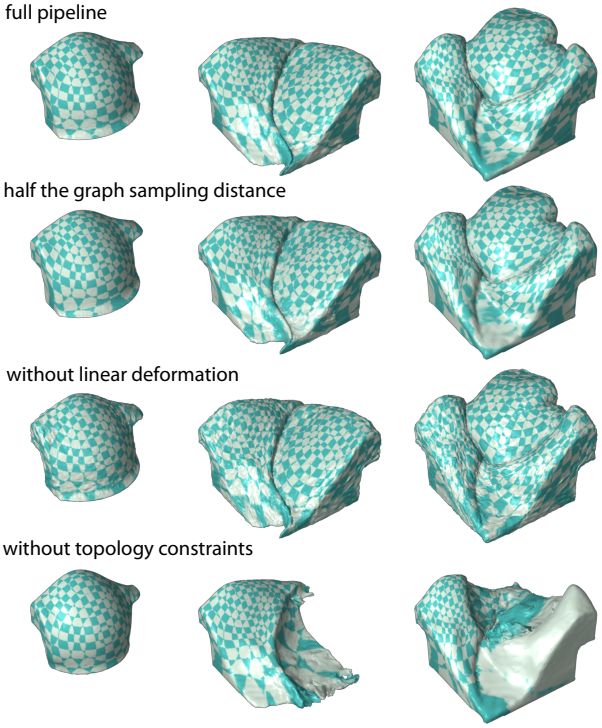

Figure 8: *Comparison between our full pipeline and leaving out individual stages our surface tracking framework.*

proach of simply projecting our tracked mesh $\mathcal{M}$ onto the input $\mathcal{T}$ each frame. Tangential drift is severe even in the case of simple translation. We compared our method to one that ignores fine-scale registration (line 6 of Algorithm 1). Since the graph-based registration works on a coarse scale and only influences vertices in $\mathcal{M}$ through linear blend weights, this modified method is unable to correctly register small features. Such errors accumulate over time, causing a rough, lumpy surface and ignoring the fine-scale details of the input. Our full algorithm clearly does not exhibit these problems, exhibiting why the fine-scale optimization in Section 4.2 is necessary.

Our tests also show that the topology constraint (§4.3, line 11 of Algorithm 1) is essential for robust tracking. The tests in our video illustrate how a method without this constraint is unable to cope with drastic changes in input topology. An obvious example in the elastic simulation is the sudden introduction of new components in later frames — when the topology constraint is turned off, the nonrigid registration algorithm was unable to recognize these components without manually creating a template. Another important feature of the topology constraint is that it acts as a convenient fail-safe. Should the registration routine fail to fully conform to the target shape, the topology constraint fills in regions of mismatched geometry. As a result, our full algorithm is quite robust to poor parameter choices for the alignment, and poor alignment leads to additional re-sampling (as opposed to an unrecoverable failure).

	Visc	Splash	Morph	Perf
Vertices	60-300	280-380	77-96	60-73
Frames	400	500	100	111
Frame time	45-153s	105-220s	17-21s	97-101s
Coarse reg.	87-93%	81-89%	67-73%	86-88%
Fine reg.	3-8%	11-18%	19-23%	10-12%

Table 1: *Summery of statistics for our application examples. Vertices are listed in thousands. Time spent on mesh improvement and topology changes is negligible compared to registration, so is omitted in the table. Timings exclude file I/O.*

Non-rigid alignment (§4.2, line 6 of Algorithm 1) dominates the time complexity of our method. The sampling density of the deformation graph is the parameter that has the biggest impact on the runtime, because it dictates the number of variables in the non-linear optimization problem. We examined the sensitivity of our algorithm to different sampling densities in our video. Instead of choosing the sampling density used to generate Figure 3, we lowered the sampling distance by a factor of one half and one quarter, then re-ran our algorithm. Our video shows that these reduced sampling densities lead to increased mesh re-sampling, but the result remains similar to our high quality tracking. Conveniently, this allows us to use lowered sampling densities to get a fast approximation of our algorithm's output before committing to solving with a high sampling density.

The memory complexity of our algorithm is similarly dominated by the non-rigid alignment. However, because we only do pairwise alignment between $\mathcal{M}$ and $\mathcal{T}$, our memory consumption is independent of the length of the sequence of input data. In other words the space complexity scales with the number of vertices in the source mesh.

We have also gathered statistics for all of our application examples. We summarize these results in table 1. All measurements were all performed on a standard PC with an Intel i7-2600K processor and 16 GB of memory. We note that our implementation has not been optimized for performance and is mostly sequential.

Comparison to other methods. As detailed in section 2 the method of Stam et. al [Stam and Schmidt 2011] is significantly different from ours. While this is an admittedly biased comparison, we show how our method performs with their example of three blended blobs rotating about the origin (see Figure 10). Our algorithm explicitly solves for the globally most rigid deformation, so we obtain practically perfect tracking whereas Stam et al. show slight tangential drift and color diffusion. We imagine their problem would be exacerbated with larger time steps, while ours remains accurate.

Limitations. Our biggest limitation is the fact that we are currently limited to closed manifold surfaces due to the method we use to performing topology changes. This method assumes that for any arbitrary point in space we must unambiguously decide whether it is inside out outside the surface.

Because our method is based on shape matching, we are unable to track surfaces invariant under our energy functions; a surface with

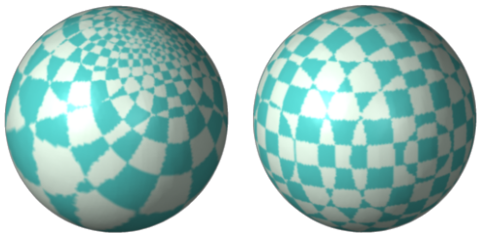

Figure 9: *The difference between projection (left) and our nonrigid registration technique (right). Simple projection causes severe distortion of the surface, while our registration reliably provides accurate correspondences.*

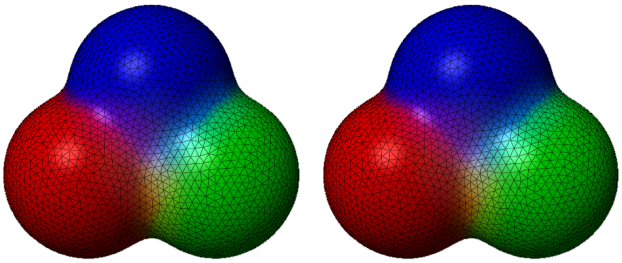

Figure 10: *Stam and Schmidt introduced shape as a benchmark for evaluating the accuracy of an implicit surface tracking algorithm. After one complete rotation, our algorithm's output (right) is virtually identical to the analytical solution (left).*

no significant geometric features (like a rotating sphere) will not be tracked accurately. However, it would be easy to augment our method with additional priors such as velocity information in order to handle such featureless cases.

7 Discussion

Relying exclusively on the geometry, our proposed framework for tracking topology evolving surfaces faithfully establishes consistent correspondences throughout entire sequences for arbitrary input mesh animations. It naturally converts any sequences of deforming implicit surfaces to a topology aware dynamic mesh data structure, making it a powerful and general tracking tool for a wide variety of examples ranging from pre-simulated fluids, performance capture data, to topology changing morphings. While exciting new applications (such as bi-directional texture tracking, wave simulations as post-processing, and performance data analysis) are made possible, we have shown that a simple combination of non-rigid registration, mesh improvement, and topology handling is not only effective in propagating correspondences through topology varying surfaces, but also, permits unrestricted shape variations in the input data (e.g., variations in surface area, volume, incoherent shape deformation).

Since our tracking approach is sequential and not relying on higher level deformation priors, we do not guarantee drift free tracking. For purposes such as tracking extended performance capture recordings, dynamic body shape statistics and elastic deformation models could be incorporated to prevent accumulations of tracking errors. Nevertheless, our performance capture example did not exhibit any noticeable drifts when propagating the texture from the first frame to the end despite the drastic topology variations and large deformations in the input data.

Future Directions We wish to explore several directions to resolve the aforementioned limitations. To expand the applicability of our approach, it would be desirable handle non-closed manifold such as dynamic 3D scans obtained from a single view sensor. The ability to parameterize dynamic surface with varying topology would also allows us to design novel texture synthesis strategies and extend our fine-scale wave simulation to not only use topology information to spawn waves. In particular, our method has the potential to use arbitrary criteria to introduce or remove wave energies which opens the door for improved artistic controls for dynamic surface textures. We believe that our tracking framework can also serve as tool for accurate validations of physical models used in fluid simulations when processing data captured from the physical world. Finally, with the increasing availability of 3D sensing technologies, we plan to investigate its effectiveness for the analysis and collection of human motions without templates.

Acknowledgements

Acknowledgements removed for peer review

References

BARGTEIL, A., GOKTEKIN, T., O'BRIEN, J., AND STRAIN, J. 2006. A semi-lagrangian contouring method for fluid simulation. *ACM Transactions on Graphics (TOG) 25*, 1, 19–38.

BARGTEIL, A., SIN, F., MICHAELS, J., GOKTEKIN, T., AND O'BRIEN, J. 2006. A texture synthesis method for liquid animations. In *Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation*, Eurographics Association, 345–351.

BOTSCH, M., KOBBELT, L., PAULY, M., ALLIEZ, P., AND LEVY, B. 2010. *Polygon mesh processing*. AK Peters Ltd.

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

BROCHU, T., BATTY, C., AND BRIDSON, R. 2010. Matching fluid simulation elements to surface geometry and topology. *ACM Transactions on Graphics (TOG) 29*, 4, 47.

CAMPEN, M., AND KOBBELT, L. 2010. Exact and robust (self-) intersections for polygonal meshes. In *Computer Graphics Forum*, vol. 29, Wiley Online Library, 397–406.

CHANG, W., AND ZWICKER, M. 2011. Global registration of dynamic range scans for articulated model reconstruction. *ACM Transactions on Graphics, to appear 30*, 3.

CHANG, W., LI, H., MITRA, N. J., PAULY, M., AND WAND, M. 2010. Geometric registration for deformable shapes. In *Eurographics 2010: Tutorial Notes*.

DINH, H., YEZZI, A., TURK, G., ET AL. 2005. Texture transfer during shape transformation. *ACM Transactions on Graphics 24*, 2, 289–310.

DU, J., FIX, B., GLIMM, J., JIA, X., LI, X., LI, Y., AND WU, L. 2006. A simple package for front tracking. *Journal of Computational Physics 213*, 2, 613–628.

ENRIGHT, D., MARSCHNER, S., AND FEDKIW, R. 2002. Animation and rendering of complex water surfaces. *ACM Transactions on Graphics (TOG) 21*, 3, 736–744.

JIAO, X. 2007. Face offsetting: A unified approach for explicit moving interfaces. *Journal of computational physics 220*, 2, 612–625.

KWATRA, V., ADALSTEINSSON, D., KIM, T., KWATRA, N., CARLSON, M., AND LIN, M. 2007. Texturing fluids. *Visualization and Computer Graphics, IEEE Transactions on 13*, 5, 939–952.

LI, H., ADAMS, B., GUIBAS, L. J., AND PAULY, M. 2009. Robust single-view geometry and motion reconstruction. *ACM Transactions on Graphics (Proceedings SIGGRAPH Asia 2009) 28*, 5.

LI, H., LUO, L., VLASIC, D., PEERS, P., POPOVIĆ, J., PAULY, M., AND RUSINKIEWICZ, S. 2012. Temporally coherent completion of dynamic shapes. *ACM Transactions on Graphics 31*, 1 (January).

MITRA, N. J., FLORY, S., OVSJANIKOV, M., GELFAND, N., GUIBAS, L., AND POTTMANN, H. 2007. Dynamic geometry registration. 173–182.

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

OSHER, S., AND FEDKIW, R. 2003. *Level set methods and dynamic implicit surfaces*, vol. 153. Springer Verlag.

PONS, J., AND BOISSONNAT, J. 2007. Delaunay deformable models: Topology-adaptive meshes based on the restricted delaunay triangulation. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, IEEE, 1–8.

SHARF, A., ALCANTARA, D. A., LEWINER, T., GREIF, C., SHEFFER, A., AMENTA, N., AND COHEN-OR, D. 2008. Space-time surface reconstruction using incompressible flow. *ACM Trans. Graph. 27* (December), 110:1–110:10.

STAM, J., AND SCHMIDT, R. 2011. On the velocity of an implicit surface. *ACM Trans. Graph 30*, 3, Article 21.

SÜSSMUTH, J., WINTER, M., AND GREINER, G. 2008. Reconstructing animated meshes from time-varying point clouds. 1469–1476.

TEVS, A., BERNER, A., WAND, M., IHRKE, I., BOKELOH, M., KERBER, J., AND SEIDEL, H.-P. 2012. Animation cartography - intrinsic reconstruction of shape and motion. *ACM Transactions on Graphics, TOG*, to appear.

THÜREY, N., WOJTAN, C., GROSS, M., AND TURK, G. 2010. A multiscale approach to mesh-based surface tension flows. *ACM Transactions on Graphics (TOG) 29*, 4, 48.

WAND, M., JENKE, P., HUANG, Q., BOKELOH, M., GUIBAS, L., AND SCHILLING, A. 2007. Reconstruction of deforming geometry from time-varying point clouds.

WAND, M., ADAMS, B., OVSJANIKOV, M., BERNER, A., BOKELOH, M., JENKE, P., GUIBAS, L., SEIDEL, H.-P., AND SCHILLING, A. 2009. Efficient reconstruction of nonrigid shape and motion from real-time 3d scanner data. *ACM Trans. Gr. 28*, 2 (Apr.), 15.

WANG, H., LIAO, M., ZHANG, Q., YANG, R., AND TURK, G. 2009. Physically guided liquid surface modeling from videos. *ACM Trans. Graph. 28* (July), 90:1–90:11.

WOJTAN, C., THÜREY, N., GROSS, M., AND TURK, G. 2009. Deforming meshes that split and merge. In *ACM Transactions on Graphics (TOG)*, vol. 28, ACM, 76.

WOJTAN, C., THÜREY, N., GROSS, M., AND TURK, G. 2010. Physics-inspired topology changes for thin fluid features. *ACM Transactions on Graphics (TOG) 29*, 4, 50.

WOJTAN, C., MÜLLER-FISCHER, M., AND BROCHU, T. 2011. Liquid simulation with mesh-based surface tracking. In *ACM SIGGRAPH 2011 Courses*, ACM.

YU, J., WOJTAN, C., TURK, G., AND YAP, C. 2012. Explicit mesh surfaces for particle based fluids. *EUROGRAPHICS 2012 30*, 41–48.

ZAHARESCU, A., BOYER, E., AND HORAUD, R. 2007. Transformesh: a topology-adaptive mesh-based approach to surface evolution. In *Proceedings of the 8th Asian conference on Computer vision-Volume Part II*, Springer-Verlag, 166–175.

ZHENG, Q., SHARF, A., TAGLIASACCHI, A., CHEN, B., ZHANG, H., SHEFFER, A., AND COHEN-OR, D. 2010. Consensus skeleton for non-rigid space-time registration. *Computer Graphcis Forum (Special Issue of Eurographics) 29*, 2, 635–644.