EUROGRAPHICS 2008 / G. Drettakis and R. Scopigno
(Guest Editors)

Volume 27 (2008), Number 3

Fluid in Video: Augmenting Real Video with Simulated Fluids

Vivek Kwatra[†1,2] Philippos Mordohai[‡1,3] Rahul Narain[§1] Sashi Kumar Penta[¶1] Mark Carlson[‖4]
Marc Pollefeys[**1,5] Ming C. Lin[††1]

[1]Department of Computer Science, University of North Carolina at Chapel Hill [2]Google, Inc.
[3]Department of Computer and Information Science, University of Pennsylvania [4]Walt Disney Animation Studios [5]ETH Zurich

Abstract

We present a technique for coupling simulated fluid phenomena that interact with real dynamic scenes captured as a binocular video sequence. We first process the binocular video sequence to obtain a complete 3D reconstruction of the scene, including velocity information. We use stereo for the visible parts of 3D geometry and surface completion to fill the missing regions. We then perform fluid simulation within a 3D domain that contains the object, enabling one-way coupling from the video to the fluid. In order to maintain temporal consistency of the reconstructed scene and the animated fluid across frames, we develop a geometry tracking algorithm that combines optic flow and depth information with a novel technique for "velocity completion". The velocity completion technique uses local rigidity constraints to hypothesize a motion field for the entire 3D shape, which is then used to propagate and filter the reconstructed shape over time. This approach not only generates smoothly varying geometry across time, but also simultaneously provides the necessary boundary conditions for one-way coupling between the dynamic geometry and the simulated fluid. Finally, we employ a GPU based scheme for rendering the synthetic fluid in the real video, taking refraction and scene texture into account.

1. Introduction

Augmentation of real world scenes with synthetic objects is a problem of significant interest for Computer Graphics (CG) as well as other research disciplines. Many special effects involve combining CG elements with real footage. Synthetic objects rendered onto real video have also been used to enhance visualization for engineering and medical applications. Most research in this area has primarily focused on using either static or rigidly moving objects as CG elements to be inserted into the scene. In contrary, we are interested in exploring the augmentation of real scenes with fluids. By being completely non-rigid, fluids pose very different challenges from those posed by rigid objects. Registration is the most important problem with rigid objects. In addition to the registration problem, fluids may flow freely anywhere in the scene and hence a much more complete reconstruction of the scene is necessary. This requirement makes insertion of CG fluid into a real video considerably more challenging, especially if the scene is dynamic in nature.

Figure 1: *An augmented video frame with CG fluid generated by our system*

† e-mail: kwatra@cs.unc.edu
‡ e-mail: mordohai@cs.unc.edu
§ e-mail: narain@cs.unc.edu
¶ e-mail: sashi@cs.unc.edu
‖ e-mail: mark.t.carlson@gmail.com
** e-mail: marc@cs.unc.edu
†† e-mail: lin@cs.unc.edu

Main Results: In this paper, we present a framework for inserting fluid phenomena, such as water and smoke, into a real dynamic scene, given a binocular (stereo-pair) video of the scene. We also demonstrate our technique on multiple-camera video data where visual hull computation can be used to obtain the foreground objects. We reconstruct the scene geometry in a manner suitable for interfacing and coupling with a fluid simulator. We allow one-way coupling from the video to the fluid, i.e. moving objects in the scene affect the behavior of the fluid but not vice-versa. In order for this coupling to be effective and for the results to look plausible, the surface of object or objects has to be closed and thus a reasonable approximation for the invisible parts has to be estimated.

Figure 2 illustrates the key steps of our pipeline. Given a binocular video sequence as input, we can reconstruct the visible surfaces in each frame using stereo vision methods. We can also process surfaces in other forms, such as point clouds, or implicit representations, such as binary indicator functions or distance fields. In these cases, the foreground extraction and depth map reconstruction steps are omitted. In general, given a binocular video sequence, we reconstruct a colored 3D model of the background from a few empty frames, which stays constant throughout the sequence. We also use this background frame to aid in the segmentation of the dynamic foreground objects in the scene, which are then used for stereo depth map reconstruction and optic flow computation. Foreground extraction is useful for two reasons. Firstly, it allows us to speed up the reconstruction of the 3D model for every frame by constraining the stereo algorithm to only use foreground objects. Secondly, it helps increase the accuracy of 2D optic flow computation, which is necessary for velocity estimation. The foreground depth map only provides information about the visible surface(s), which need to be converted into a *complete* 3D object representation for it to interact with the animated objects (e.g. fluids). To achieve this goal, we use a surface completion technique that combines the current depth map with predicted estimates for foreground geometry. These predictions, in turn, are obtained by transporting the geometry in the previous frame by its velocity. The velocity field itself is computed by combining optic flow with the foreground depth maps, which gives us partial scene flow (for the visible regions). We obtain the velocity field for the entire foreground object(s) using a novel formulation for *velocity completion* that keeps connected objects as locally rigid as possible.

Once we have the completed geometry and velocity information for each frame, we can design CG animations that interact with the real scene, as we demonstrate by simulating fluids in real video. Since we only have color information for the viewpoints of the cameras, the rendered output is constrained to be for one of the original viewpoints. Finally, we employ a GPU based scheme for rendering fluids in the video. We show results using several fluids with different properties. Our rendering takes refraction, reflection and partial occlusion from the fluid into account and uses the texture of the scene to generate convincing results.

To summarize, the key results of our work presented in this paper include:

- Novel algorithms for computing reconstructed 3D geometry and velocity fields for the hidden geometry of the scene by combining accurate estimates for visible regions with appropriate rigidity and temporal constraints;
- An effective rendering mechanism that combines texture and geometry from the video with CG fluids to synthesize the appearance of fluid phenomena in real video;
- An integrated system that generates temporally consistent geometry and velocity fields from input video data that contain only partial geometry and velocity information, which can then be augmented with a computer animation system, such as a fluid simulator, to easily create interesting visual effects with little human intervention.

We demonstrate our integrated system on two sequences of video footage with different fluid phenomena as shown in Fig. 1 to illustrate its potential. Our system enables a seamless integration of virtual CG elements into a real video sequences, leading to many possible novel applications in engineering and scientific visualization.

Organization: The rest of the paper is organized as follows. Section 2 briefly surveys related work. We describe our 3D geometry reconstruction algorithm in Section 3 and our novel velocity field computation and completion approach in Section 4. Section 5 describes the interaction between the reconstructed geometry and the fluid simulator. We present a fast image-based method for rendering synthetic fluids in a pre-recorded video sequence in Section 6 and demonstrate the resulting system in Section 7. We finally conclude and discuss several possible future research directions.

2. Background

Rendering synthetic simulations into real world scenes is an important application of both computer graphics and computer vision. Thus, our work draws upon research in both fields. Computer vision techniques can be used to reconstruct 3D models of real world scenes, whereas computer graphics can be used to render these reconstructed 3D models, as well as the synthetic models.

The reconstruction of 3D models is the focus of stereo vision, which is among the most active areas in computer vision. Due to lack of space we refer readers to [SSZ02, SCD*06]. We are interested in the reconstruction of dynamic scenes from multiple video streams captured by stationary calibrated cameras. Zitnick et al. [ZKU*04] present an approach for video view interpolation that generates realistic videos with varying viewpoints, even though

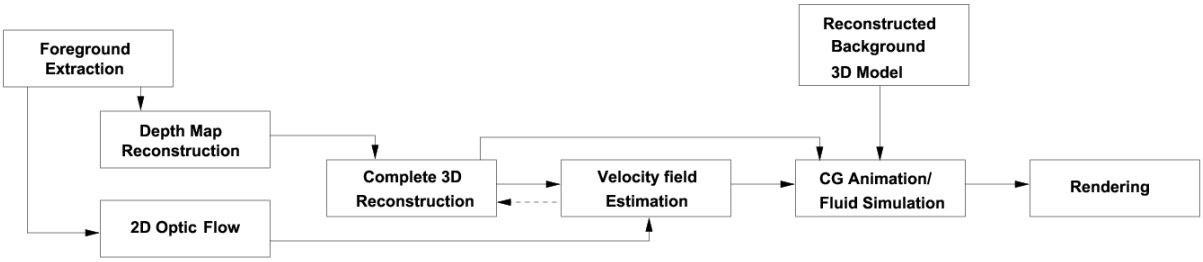


Figure 2: *System Overview Diagram*

each frame is processed separately. Methods that enforce temporal consistency have been reported in the literature [VBR*05, CK02, GILM07].

Vedula et al. [VBR*05] estimate what is termed *3D scene flow* which is the 3D equivalent of optic flow. Scene flow is a dense motion field for all surface points of the scene from frame to frame. Its projection on one of the images gives optic flow. Our approach is similar in that we also compute optic flow [BA96] to infer approximate 3D velocities of scene points, but scene flow by itself does not satisfy our requirement for closed surfaces if the cameras do not view the objects completely. Consequently, we interpolate missing velocities using a novel local rigidity constraint, which is similar in spirit to that of Alexa et al. [ACOL00] and Igarashi et al. [IMH05] for as-rigid-as-possible shape interpolation and manipulation, respectively. While those techniques rely on a one-to-one mapping between pairs of shapes, our method directly works with velocity vectors.

Snavely et al. [SZKC06] approached a problem similar to ours. Given 2.5-D videos, in which depth is computed using structured light, they apply 3D effects to achieve non-photorealistically rendered videos. The first steps of their algorithm are similar to ours: foreground/background separation, stereo reconstruction and detection of temporal correspondences. To obtain the desired quality for their application, they simplify some of the tasks by using black background to enable automatic segmentation and by using an active sensor (structured light) for the reconstruction. In addition to facing the same challenges, our application requires complete surfaces, even if they are not visible, to avoid implausible interactions between the objects and the fluid.

Surface completion is an inherent ability of the human visual system that is hard to replicate in an artificial system. According to researchers in psychology, neuroscience and computer vision, completion relies on various cues, which may be local (good continuation), global (symmetry), and prior knowledge [BF05]. Here, we do not attempt to handle complex occlusion, but rather to produce plausible completions for 2.5-D surfaces produced by stereo reconstruction. Our approach is based on the method of Kazhdan et al. [KBH06] who cast surface reconstruction from an oriented set of points as a spatial Poisson problem. They proposed an adaptive multi-scale algorithm that infers a plausible indicator function, which is 1 inside the volume and 0 outside. This method is robust to noise and thus suitable for our data.

Fluid simulation and various related natural phenomena have received much attention recently. Foster and Metaxas [FM96] were among the first to present the use of the full 3D Navier-Stokes differential equations for generating fluid animations in computer graphics. The "stable fluids" method of Stam [Sta99] introduced stable semi-Lagrangian advection combined with an implicit viscosity solver to arrive at a completely stable method, more amenable to use in animation. Level set methods [FSJ01, FF01, EMF02, LGF04, CMT04] have been used very successfully for liquid surface tracking. Other mesh-free methods [SF95, DG96, MCG03, Liu02, LL03] have also been proposed to simulate fluid dynamics. We refer the readers to more detailed surveys in [LY05, SSK05].

With regard to augmenting real worlds with fluid or other physical simulation, Allard and Raffin [AR06] have developed a Virtual Reality system for this purpose. Their focus, however, is real time performance as opposed to high quality simulation, while our goal is high quality reconstruction and animation using as little as two cameras. Khan et al. [KRFB06] present a method for replacing the material of scene objects with completely different materials, including transparent and translucent ones. They argue that the user tolerates certain inaccuracies in such a system. This is true for our application as well. Inaccuracies that cannot be tolerated, however, include the location and velocity of object boundaries that interact with the synthetic fluid, which will be addressed in this paper. The work of Jancène et al. [JNP*95] is close to ours in that they also models objects from video for Augmented Reality applications. However, they insert solid CG objects into the scene as opposed to fluids, and hence deal with somewhat different issues.

3. Geometry Reconstruction

In this section we present our approach for recovering 3D geometry from the captured videos using two stationary cameras. Our reconstruction strategy is to treat the static (background) and dynamic (foreground) parts of the scene separately for multiple reasons. We begin by capturing a few frames of the static background without any foreground to

initialize our background model. It includes both appearance and geometry, which we obtain by running our stereo algorithm on these images. The appearance model can be used to segment the foreground in all subsequent frames. After segmentation, we perform stereo reconstruction on the segmented foreground only using an approach based on graph-cuts [BVZ01] due to the high quality results it produces in settings such as ours, that is in computing open surfaces (2.5-D depth maps). We also estimate the optic flow on the segmented images and combine the output with the geometry to obtain 3D velocity estimates that are used to drive the interaction between the reconstructed geometry and the fluid.

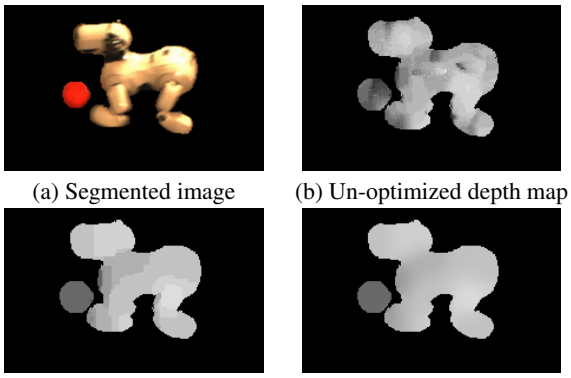
3.1. Background Modeling

The use of a background model that encompasses both appearance and geometry brings about several advantages. Regarding the background itself, it results in a temporally consistent reconstruction, as opposed to reconstructing the background independently in each frame, which besides being redundant, may cause artifacts in the simulation. Slight miss-estimations in shape from frame to frame can cause jittering which in turn will affect the fluid. Velocities do not have to be computed for the static parts of the scene since they are always zero. Thus the geometry of the background provides boundary conditions to the fluid simulation that remain constant for all frames.

The advantages of separate foreground processing include improvements in both efficiency and quality. Since high-quality stereo methods are relatively expensive, we gain a lot in terms of speed by applying the graph-cut algorithm to the foreground only. Besides the reduced number of pixels that are processed, the number of potential correspondences for each pixel are considerably fewer and correspondence ambiguity is reduced. The same apply to the optic flow computation. An additional benefit, shown in [TSK01], is that after segmentation we can produce depth maps with exceptionally sharp object boundaries. This aspect is important since the object boundaries as seen in the reference camera are the interfaces with the fluid where disturbing artifacts would appear if the geometry was inaccurate.

3.2. Foreground Extraction

Several methods for extracting the foreground from video sequences have been reported in the literature [RKB04, SZTS06, KCB*06]. We use the one of Sun et al. [SZTS06] which estimates Gaussian mixture models for the background and foreground color distributions and a per-pixel appearance model of the static background. The optimal segmentation is computed using graph cuts with two labels and a smoothness term that attenuates the effect of background edges. The attenuation of background edges enforces smoothness on the background and allows the edges between the foreground and background to guide the segmentation.



(a) Segmented image (b) Un-optimized depth map

(c) Output of graph-cuts (d) Filtered depth map

Figure 3: *Steps of stereo reconstruction for a frame of the video sequence of an AIBO robot dog: (a) segmented image from the left camera; (b) un-optimized depth map in which each pixel is assigned the disparity with the minimum data term; (c) the result after graph-cuts in which noise has been significantly reduced; (d) final depth map after filtering in which disparity varies smoothly over the entire AIBO and is not piece-wise constant as in (c).*

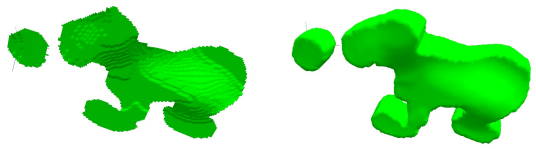
Since our cameras may not be photometrically calibrated, we use separate models for the left and right video stream.

3.3. Stereo Correspondence

We used the graph-cut based stereo approach of Boykov et al. [BVZ01] to compute depth maps for the background once, as well as for the foreground of each frame. Graph-cut methods aim at the minimization of an energy function that has two terms: a *data term* and a *smoothness term*. The former is a function of the cost of assigning a certain label to each pixel and the second is a function of the dissimilarity between labels of two adjacent pixels. The smoothness cost penalizes the labeling if neighboring pixels have different labels. The output of the graph cut is an assignment of a label (disparity) to each node (pixel) of the graph that attains a strong local minimum of the energy. Figure 3(b) shows the depth map for a frame of the AIBO sequence (see Section 7 for more details and results) if we select the minimum cost of the data term for each pixel. The number of errors is reduced in the graph cut solution (Fig. 3(c)).

The disparity map is relatively accurate and noise-free, but suffers from bias towards fronto-parallel surfaces that is inherent in graph cuts. To reduce this bias we iteratively apply a simplified bilateral filter (typically 3×3) on the depth map to obtain a smoother surface. The filter is designed similar to [TM98] so that it does not blend disparities across pixels that are more than a few disparity levels apart. In all our experiments, pixels do not affect each other if their disparities differ by 3 levels or more.

3D reconstruction of the visible parts of the surfaces is



(a) 2.5-D surface (b) Completed surface

Figure 4: *2.5-D surface for a video frame of the AIBO sequence and 3D surface completion (see Section 3.4)*

performed by a module that computes the depth of each pixel and constructs a mesh. The vertices of the mesh for the foreground are computed by triangulating the rays of corresponding pixels. The triangles of the mesh are formed by linking vertices that project to adjacent pixels. Triangles are not generated if there is a depth discontinuity between any of their vertices. Furthermore, we can compute the normal for each vertex using its 3D position and neighboring vertices.

3.4. Complete 3D from 2.5-D Surfaces

While a depth map may be sufficient for creating a 3D mesh of the visible parts of the surfaces or for view interpolation, it is not sufficient for running a fluid simulation through the scene. Such an operation requires the construction of a closed, watertight geometric mesh that can be converted into a signed distance function (SDF) – our fluid simulator uses the SDF as boundary representation. Watertight geometry is necessary because the fluid may go into regions occluded in the view, resulting in unrealistic phenomena. Note that obtaining such a watertight mesh is non-trivial because the occluded portions are invisible to the camera and therefore ambiguous. Our goal is to obtain a plausible but not necessarily accurate completion of the objects. Such a surface is sufficient for our purposes since the results of the simulation are rendered from the viewpoint of one of the cameras.

Here we adapt the algorithm of Kazhdan et al. [KBH06] that reconstructs a closed surface given a set of sparse oriented points, potentially corrupted by significant amounts of noise. The unknown surface that bounds the volume can be represented as the locus of non-zero gradients of a binary indicator function, which is 1 inside the object and 0 outside. This results in a spatial Poisson problem solved in an adaptive multi-scale fashion. Our experiments have shown that even in the case of a 2.5-D reconstruction, where all points are on one side of the object, this algorithm is able to produce a plausible shape that is consistent with the visible parts of the surface. To aid the completion we add a few points, typically less than 10, along the viewing lines of the silhouette of the foreground. (Viewing lines are rays from the camera center through the contour of the silhouette.) These points are added at distances that are similar to the distance between neighboring reconstructed points. Even though the thickness of the object is unknown, the inferred invisible side is important for effective simulations.

We typically perform the completion step in this fashion to obtain a closed surface for the first frame of the sequence. Figure 4 shows an example of a 2.5-D surface and the 3D completion obtained using only the points from the reconstruction of a single frame and the additional points on the viewing lines. After the first frame, this completed surface is combined with velocity information and subsequent partial surfaces to obtain similar surfaces for the following frames; this procedure is described in the next section.

4. Velocity Field Computation

For the simulated fluid to interact realistically with the dynamic objects in the scene, it is necessary that the fluid follows the scene objects closely. This requires the computation of a velocity field on the surface of the object, which can then be used as boundary condition for the fluid simulator. We start by computing the 2D optic flow field from segmented images. This flow is then converted to 3D scene flow at each visible surface point using the depth map. Using these initialized velocities at visible points as constraints, we then solve for the velocities at all points – visible and invisible – of the surface. We achieve this by solving a novel optimization problem that keeps the surface velocity field as *locally* rigid as possible. We represent the surface as a mesh, and the 3D velocity is computed at every vertex. While accuracy does not need to be very high for the invisible parts of the surface, which remain invisible in the rendering, the completed surface and its velocity should be plausible as well as temporally coherent so that they do not cause unexpected or unrealistic effects when interacting with the fluid. The required processing steps are described below in detail. If instead of binocular video, the input came in the form of a point cloud or a visual hull, these steps would be the first stages of processing.

4.1. Partial Scene Flow

We start velocity computation by determining optic flow for the foreground. The use of a segmented foreground has two advantages: it restricts processing to the dynamic parts of the scene and helps the optic flow algorithm to disambiguate the flow near object boundaries. Consequently, we obtain a flow field that is much more accurate than the one achieved by running optic flow on the full frame. We use the technique of [BA96] for computing optic flow. Note that since we have multiple views, the flow computation is run for each camera.

Optic flow computation gives us velocity information in the 2D image plane of each camera. The next step is to project this 2D velocity into 3D by combining flow and stereo information from each camera. There exist prior approaches for performing this operation. For example, [VBR*05] describe different mechanisms for combining optic flow from multiple cameras. To obtain the 3D velocity at a given vertex of the reconstructed surface mesh, we first

predict its 3D velocity using the optic flow for each camera in which it is visible: for every such camera, we compute the 2D projection of the vertex on the camera plane, and transport this 2D location along the optic flow vector to obtain its new location in the next frame. The depth at this transported location gives us the 3D position of the vertex in the next frame, and effectively determines the 3D velocity (displacement) of the vertex. These predicted velocities from all cameras in which the vertex is visible are then combined through a weighted mean to obtain the final 3D velocity of the vertex.

4.2. Velocity Completion

The scene flow computed above using 3D projection of optic flow determines the velocity only for visible vertices. Additionally, near depth discontinuities, either the vertices may not be visible in both the current and the next frame, or the optic flow computation may not be reliable. Consequently, the appropriate velocity is not always available at all vertices of the mesh. Given this partial scene flow information for the foreground, the next step is to compute the velocity at all vertices of the mesh.

We propose a novel approach for *completing* the surface velocity field at all vertices of the foreground mesh from the partial scene flow computed above. We postulate a local rigidity assumption on the motion of the mesh, i.e., nearby points on the mesh should move as rigidly as possible. Note that we cannot make a global rigidity assumption, since the foreground may consist of many different objects, some of which may be articulated. However, it is reasonable to assume that most regions of the surface that are in close proximity will move rigidly with respect to each other. This assumption can be formulated into a constraint as follows:

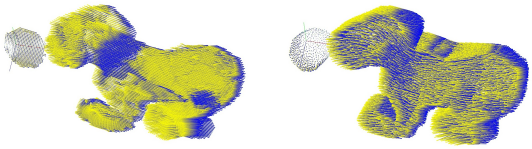
Let $\mathbf{p}_a$ and $\mathbf{p}_b$ be two points on the surface moving with velocities $\mathbf{v}_a$ and $\mathbf{v}_b$. A rigid motion of the line segment, $\mathbf{d} = \mathbf{p}_b - \mathbf{p}_a$, joining them would imply that its length $|\mathbf{d}|$ stays constant. That will happen only if the relative velocity between the two points, $\mathbf{v_d} = \mathbf{v}_b - \mathbf{v}_a$, is orthogonal to $\mathbf{d}$. Hence, the local rigidity constraint between the two points can be written as:

$$\mathbf{d} \cdot \mathbf{v_d} = 0.$$

However, this constraint may not be exactly satisfied everywhere for a set of velocity vectors that are part of the partial scene flow computed above. A velocity field that minimizes its deviation from this constraint can be obtained by minimizing the following cost function:

$$C(\mathbf{v}) = \sum_{\mathbf{d}} \mathbf{v_d}^T \mathbf{d}\mathbf{d}^T \mathbf{v_d}, \qquad (1)$$

treating the partial scene flow velocity vectors, where available, as *boundary conditions*. $\mathbf{d}$ iterates over all edges in the polygonal mesh representing the foreground surface. Equation 1 is a type of least squares formulation for the velocity field. It requires solving a linear system of equations, for which we use the Preconditioned Conjugate Gradient method. An example of the estimated velocity vectors for the visible parts of the geometry can be seen in Fig. 5(a). The completed velocities for the entire surface are shown in Fig. 5(b). Effectively, our algorithm computes a coherent velocity field for the object by propagating information from visible vertices to invisible ones, through the edge paths that connect them in the mesh.



(a) Partial scene flow (b) Scene flow for completed object

Figure 5: *3D velocity vectors for the 2.5-D surface for a frame of the AIBO sequence and the result of our velocity completion algorithm as described in Section 4.2. The ball is static and thus has zero-length velocity vectors.*

4.3. Shape Prediction

The output of the velocity completion step is a 3D mesh with a velocity vector associated with each vertex. One could repeat the procedure of Sections 3.4 through 4.2 for each frame. The results, however, could contain large inconsistencies between frames. To maintain temporal consistency we use the shape and velocity estimates of the current frame combined with the partial stereo reconstruction of the following frame to estimate the shape of the object or objects in the following frame. Specifically, we apply the surface completion algorithm of Section 3.4 simultaneously on two sets of input points:

- The first set of points is obtained by transporting the vertices of the current frame using their 3D velocities as estimated in Section 4.1 and completed in Section 4.2.
- The second set of points is obtained by stereo as in Section 3.3 and covers only the visible parts of the surface.

The Poisson surface completion algorithm of Section 3.4 is applied to the merged point cloud containing both sets of points. This procedure has the effect of combining the instantaneous measurement of the surface obtained via stereo, with its shape predicted from the previous time step. Consequently, the portions of the surface visible to the stereo algorithm in the current frame have a crisp appearance, while the missing parts are synthesized in a fashion that maintains spatio-temporal continuity of the surface. Finally, we use the foreground silhouettes to carve away parts of the

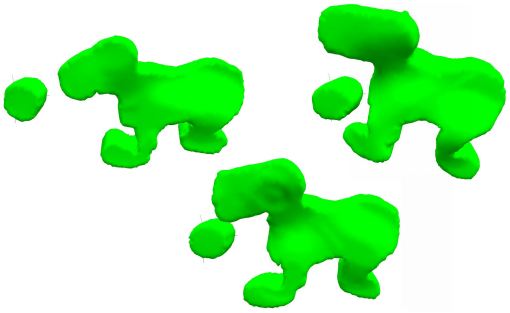


Figure 6: *Screenshots of the completed objects in the scene taken from novel viewpoints. The frames are not consecutive.*

surface that project on background pixels. This step is necessary to obtain accurate boundaries, which are crucial for convincing interaction with the fluid. Another important requirement is consistency in time to avoid jitter in the simulation. The shape prediction step described in this section maintains consistency in the foreground model. The background model, on the other hand, is kept constant throughout the simulation to prevent it from causing jitter. Figure 6 contains screenshots of the completed 3D objects from a viewpoint different than those of the cameras. Note that our prediction-completion algorithm is even able to overcome failures in stereo computation, which can happen occasionally for some frames if the cameras are not fully synchronized. We have in fact used it to interpolate the shape for several unsynchronized frames in the AIBO sequence (see 7 for results).

5. Interaction with Fluid Simulation

Once we have processed the video data to obtain completed shapes and velocities of the dynamic objects in the scene, we are ready to insert fluid behavior into the scene. To this end, we use the processed data to generate boundary conditions for our fluid simulator, which consequently ensures a one-way coupling from the video to the fluid, i.e., the geometry and motions present in the video affect the behavior of the fluid, but not vice-versa.

We use a grid-based Eulerian approach for fluid simulation, which is similar in spirit to the method of Stam [Sta99], but has many of the improvements proposed by later authors, such as [CMT04]. The fluid simulator discretizes the 3D space into a uniform grid of cells. It then solves for the fluid velocity in each of these cells at every time step of the simulation. To construct a fluid animation that properly interacts with the video, we need to discretize the shape and velocity of the objects in the video onto the same grid on which the simulation is performed. In the previous steps, we compute a boundary (triangle-mesh) representation for the geometry of the foreground and background objects in the scene. For simulation, this boundary representation is converted into a signed distance function using the Fast Marching method [Set98]. This conversion is performed just once for the (stationary) background. The foreground, on the other hand, is discretized for every time step. Once the geometry is discretized on the grid, the velocity from the surface points can be *extended* onto grid locations by evolving them along the surface in a way that preserves the signed distance function (see [AS99] for details).

Another aspect of simulation is the specification of the fluid in the domain. For example, one could start with an initial fluid volume that then evolves under the simulation. One could also specify emitters and drains for the fluid in the scene. Our simulation API provides an interface for performing these initializations – currently we interact with the API programmatically but it is foreseeable to build a user-interface around it that would further ease the task of authoring simulations.

6. Rendering

The visual plausibility of the rendered result depends greatly on the accuracy of scene silhouettes. If the silhouettes are not correct, the fluid will not be occluded properly by objects in the scene, and such a visual artifact is very noticeable. On the other hand, the variation of depth within an object silhouette is less visually important. Therefore, for the purpose of rendering, we use the 3D information of the scene directly from the depth map rather than the volumetric model generated for simulation.

Smoke rendering is done using a standard ray marching approach [FSJ01]. For rendering liquids, we use ray tracing to render them into the scene with correct reflections and refractions. The intersection test of a ray with the scene's depth map is performed by comparing the projective depth of a point on the ray with the value of the depth map at that position, and applying the bisection method. We have implemented a version of this rendering algorithm which runs on the GPU, based on recently proposed techniques such as [DW07, HQ07]. However, the presence of large depth discontinuities in real scenes can lead to prominent visual artifacts. These can be made less noticeable by biasing the ray–depth-map intersection towards the farther side of the discontinuity, so that the color chosen for the ray is taken from the background rather than a foreground object. The renderer runs in real-time on current graphics hardware.

7. Results

We demonstrate our results by simulating liquid and smoke animations within real video. We used a grid-based fluid solver for our simulations, which employed the shapes and velocities computed by our algorithm as boundary conditions. Figure 7 shows an example where a baby is crawling on a table. We drop a blob of synthetic fluid on the baby and it correctly interacts with the crawling baby. Figure 8 uses the same video as input but here water is pour-

ing via an imaginary hose on the baby. Note that for this baby sequence, visual hull based shape information was already available to us, courtesy of Brostow et al. [BESK04]. However, we still needed to perform velocity completion for proper dynamic behavior of the fluid. The visual hull also provides ground truth information. In Figure 8, we compare the hose result using visual hull based shape, and that obtained using our completion algorithm after eliminating all but two of the nine cameras used in visual hull reconstruction. The results look quite similar, which reinforces our claim that the surfaces obtained through the completion algorithm are good enough for simulation.

The complete pipeline has been demonstrated on videos featuring a Sony AIBO Entertainment Robot. The videos were captured by two synchronized 1024×768 video-cameras. The user was part of the loop in the segmentation stage, in which some adjustments were required, and also designed the fluid simulations. Figure 10 shows some frames of the AIBO interacting with different fluids. Notice how our system allows a user to edit the material properties of the fluid in addition to generating novel simulations. Here we show examples where the AIBO interacts with water and honey. In Figure 9, we show simulated smoke interacting with the AIBO. Please see the supplemental material for animated videos and more examples.

Figure 8: *Left: Visual Hull, Right: Completion*

8. Implementation Issues and Discussion

As we go through our processing pipeline, the objects in the scene undergo various transformations and change of representations. The stereo algorithm generates 2.5-D depth maps, which are represented as a triangle-mesh. This triangle-mesh gets converted into a volumetric representation by the Poisson surface completion algorithm. At this stage, we re-convert the objects into a triangle-mesh representation for velocity completion, after which the geometry is further discretized onto the volumetric fluid grid for simulation. There are two reasons for going back to the triangle-mesh representation in the velocity completion stage. Firstly, the velocity completion itself is faster and more robust on the triangle-mesh, because the rigidity constraints are propagated over a 2-D manifold as opposed to through a 3D volume resulting in a more compact system. Also, the formulation for velocity completion presented here assumes a

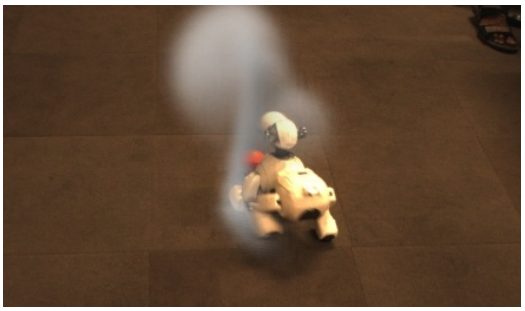


Figure 9: *AIBO interacting with smoke.*

triangle-mesh representation and would need to be modified non-trivially to make it work with a volumetric representation. Secondly, the two volumetric representations, used for completion and simulation respectively, serve different purposes and typically have different resolutions. The completion algorithm uses a multi-resolution octree-based grid, while the simulation uses a (generally coarser) uniform grid. Therefore, one needs to convert between these representations.

The Poisson completion algorithm was originally designed for scenarios where only small parts of the object are missing. In our case, the first frame provides a case where large parts of the object are missing, which can cause the algorithm to potentially generate unstable results, thus sometimes requiring manual tweaking of the completed surface to make it satisfactory. However, such processes typically only need to be done in the first frame, since in the subsequent frames the velocity based propagation algorithm generates points over most of the object, thus enabling robust completion of the surface. A promising direction for future work is to jointly optimize the shape in all frames, which should make the algorithm more robust to noise and occlusion.

Our current approach assumes local rigidity for velocity completion and does not impose any global volume-preservation constraints. In other words, the reconstructed and completed objects always project correctly onto the video frames, but they may change (generally increase slightly) their volume over time. We have not noticed any artifacts related to the volume increase in our results. However, it may be desirable to enforce the starting volume as a constraint throughout the sequence.

9. Conclusions

We have presented a framework for inserting fluid phenomena into a real dynamic scene that includes articulated objects. We reconstruct the scene geometry in a manner suitable for interfacing and coupling with a fluid simulator. In order for this coupling to be effective, the surface of the objects has to be closed and thus a reasonable approximation for the invisible parts has to be estimated. To this end we have presented novel algorithms for computing 3D geome-

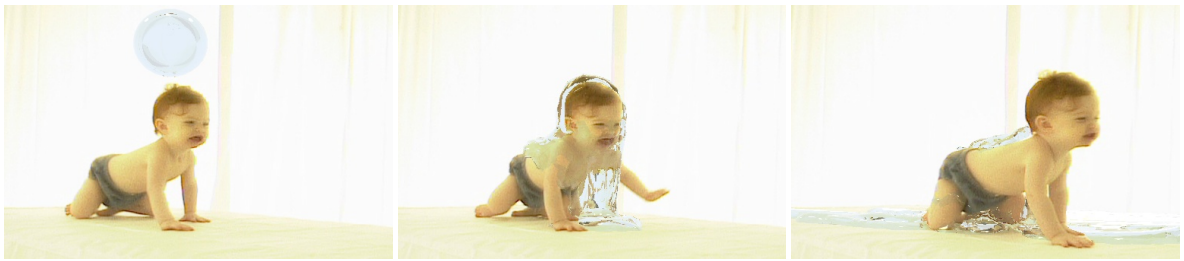


Figure 7: *Pouring synthetic water over a baby*

Figure 10: *AIBO video with various fluid examples.*

try and velocity fields for the hidden geometry of the scene by combining estimates for visible regions with appropriate rigidity and temporal constraints. We have shown that, while these estimates do not correspond to the actual object, they enable us to generate convincing visual effects with limited user input. The final stage of our system is an effective rendering mechanism that combines the texture and geometry from the video with CG fluid data to synthesize the appearance of fluid phenomena in real video. Our future work will focus on the development of an interactive tool that would make the design and testing of such simulations even easier.

Acknowledgements

This research is supported in part by ARO Contracts DAAD19-02-1-0390 and W911NF-04-1-0088, NSF awards 0400134, 0429583, and 0636208, DARPA/RDECOM Contract N61339-04-C-0043, and Intel. This work was carried out while the first and the second authors were postdoctoral research associates at the University of North Carolina, Chapel Hill. We would like to thank Gabriel Brostow for providing the data for the baby sequence, as well as Jean-Sébastien Franco for his GPU framework code.

References

[ACOL00] ALEXA M., COHEN-OR D., LEVIN D.: As-rigid-as-possible shape interpolation. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 2000), ACM Press/Addison-Wesley Publishing Co., pp. 157–164.

[AR06] ALLARD J., RAFFIN B.: Distributed physical based simulations for large vr applications. In *IEEE Virtual Reality Conference* (Alexandria, USA, March 2006).

[AS99] ADALSTEINSSON D., SETHIAN J.: The fast construction of extension velocities in level set methods. *Journal of Computational Physics 148* (1999), 2–22.

[BA96] BLACK M. J., ANANDAN P.: The robust estimation of multiple motions: Parametric and piecewise-smooth flow fields. *Computer Vision and Image Understanding 63*, 1 (1996), 75–104.

[BESK04] BROSTOW G. J., ESSA I., STEEDLY D., KWATRA V.: Novel skeletal representation for articulated creatures. In *European Conf. on Computer Vision* (2004), pp. 66–78.

[BF05] BRECKON T., FISHER R.: Amodal volume completion: 3d visual completion. *Computer Vision and Image Understanding 99*, 3 (2005), 499–526.

[BVZ01] BOYKOV Y., VEKSLER O., ZABIH R.: Fast approximate energy minization via graph cuts. *In IEEE Transactions on Pattern Analysis and Machine Intelligence 23*, 11 (2001), 1222–1239.

[CK02] CARCERONI R., KUTULAKOS K.: Multi-view scene capture by surfel sampling: From video streams to non-rigid 3d motion, shape and reflectance. *Int. Journal of Computer Vision 49*, 2-3 (2002), 175–214.

[CMT04] CARLSON M., MUCHA P., TURK G.: Rigid fluid: Animating the interplay between rigid bodies and fluid. In *SIGGRAPH* (2004).

[DG96] DESBRUN M., GASCUEL M. P.: Smoothed particles: A new paradigm for animating highly deformable bodies. In *Computer Animation and Simulation '96 (Proceedings of EG Workshop on Animation and Simulation)* (1996), Springer-Verlag, pp. 61–76.

[DW07] DAVIS S., WYMAN C.: Interactive refractions with total internal reflections. In *Graphics Interface (to appear)* (2007).

[EMF02] ENRIGHT D., MARSCHNER S., FEDKIW R.: Animation and rendering of complex water surfaces. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques* (2002), pp. 736–744.

[FF01] FOSTER N., FEDKIW R.: Practical animations of liquids. In *SIGGRAPH* (2001), pp. 23–30.

[FM96] FOSTER N., METAXAS D.: Realistic animation of liquids. *Graphical models and image processing: GMIP 58*, 5 (1996), 471–483.

[FSJ01] FEDKIW R., STAM J., JENSEN H. W.: Visual simulation of smoke. In *SIGGRAPH* (2001), pp. 15–22.

[GILM07] GOLDLUECKE B., IHRKE I., LINZ C., MAGNOR M.: Weighted minimal hypersurface reconstruction. *IEEE Transactions on Pattern Recognition and Machine Intelligence 29*, 7 (2007), 1194–1208.

[HQ07] HU W., QIN K.: Interactive approximate rendering of reflections, refractions, and caustics. *IEEE Transactions on Visualization and Computer Graphics 13*, 1 (2007), 46–57.

[IMH05] IGARASHI T., MOSCOVICH T., HUGHES J. F.: As-rigid-as-possible shape manipulation. *ACM Trans. Graph. 24*, 3 (2005), 1134–1141.

[JNP*95] JANCÈNE P., NEYRET F., PROVOT X., TAREL J.-P., VÉZIEN J.-M., MEILHAC C., VERROUST A.: RES: computing the interactions between real and virtual objects in video sequences. In *Second IEEE Workshop on Networked Realities (NR'95)* (Boston, Massachusetts (USA), 1995), pp. 27–40.

[KBH06] KAZHDAN M., BOLITHO M., HOPPE H.: Poisson surface reconstruction. In *Eurographics Symposium on Geometry Processing* (2006), pp. 61–70.

[KCB*06] KOLMOGOROV V., CRIMINISI A., BLAKE A., CROSS G., ROTHER C.: Probabilistic fusion of stereo with color and contrast for bilayer segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence 28*, 9 (2006), 1480–1492.

[KRFB06] KHAN E. A., REINHARD E., FLEMING R. W., BÜLTHOFF H. H.: Image-based material editing. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Papers* (New York, NY, USA, 2006), ACM Press, pp. 654–663.

[LGF04] LOSASSO F., GIBOU F., FEDKIW R.: Simulating water and smoke with an octree data structure. *ACM Transactions on Graphics 23*, 3 (2004), 457–462.

[Liu02] LIU G. R.: *Mesh Free Methods: Moving Beyond the Finite Element Method*, 1st ed. CRC Press, 2002.

[LL03] LIU G. R., LIU M. B.: *Smoothed Particle Hydrodynamics: A Meshfree Particle Method*. World Scientific Pub. Co., Inc., 2003.

[LY05] LIN S., YU Y.: Controllable smoke animation with guiding objects. *ACM Transactions on Graphics 24*, 1 (2005), 140–164.

[MCG03] MULLER M., CHARYPAR D., GROSS M.: Particle-based fluid simulation for interactive applications. *Proc. of ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2003).

[RKB04] ROTHER C., KOLMOGOROV V., BLAKE A.: "grabcut": interactive foreground extraction using iterated graph cuts. In *SIGGRAPH* (2004), pp. 309–314.

[SCD*06] SEITZ S. M., CURLESS B., DIEBEL J., SCHARSTEIN D., SZELISKI R.: A comparison and evaluation of multi-view stereo reconstruction algorithms. In *Int. Conf. on Computer Vision and Pattern Recognition* (2006), pp. 519–528.

[Set98] SETHIAN J.: Level set methods and fast marching methods: Evolving interfaces in computational geometry, 1998.

[SF95] STAM J., FIUME E.: Depicting fire and other gaseous phenomena using diffusion processes. In *SIGGRAPH* (1995), pp. 129–136.

[SSK05] SONG O., SHIN H., KO H.: Stable but nondissipative water. *ACM Transactions on Graphics 24*, 1 (2005).

[SSZ02] SCHARSTEIN D., SZELISKI R., ZABIH R.: A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *Int. Journal of Computer Vision 47*, 1 (2002).

[Sta99] STAM J.: Stable fluids. In *SIGGRAPH* (1999), pp. 121–128.

[SZKC06] SNAVELY N., ZITNICK C., KANG S., COHEN M.: Stylizing 2.5-d video. In *International Symposium on Non-Photorealistic Animation and Rendering* (2006), pp. 63–69.

[SZTS06] SUN J., ZHANG W., TANG X., SHUM H.-Y.: Background cut. In *European Conf. on Computer Vision* (2006), pp. 628–641.

[TM98] TOMASI C., MANDUCHI R.: Bilateral filtering for gray and color images. In *Int. Conf. on Computer Vision* (1998), pp. 839–846.

[TSK01] TAO H., SAWHNEY H., KUMAR R.: Dynamic depth recovery from multiple synchronized video streams. In *Int. Conf. on Computer Vision and Pattern Recognition* (2001), pp. I:118–124.

[VBR*05] VEDULA S., BAKER S., RANDER P., COLLINS R., KANADE T.: Three-dimensional scene flow. *IEEE Transactions on Pattern Recognition and Machine Intelligence 27*, 3 (2005), 475–480.

[ZKU*04] ZITNICK C. L., KANG S. B., UYTTENDAELE M., WINDER S., SZELISKI R.: High-quality video view interpolation using a layered representation. In *SIGGRAPH* (2004).