Realtime Video Based Water Surface Approximation

Chuan Li, Martin Shaw, David Pickup, Darren Cosker, Phil Willis and Peter Hall

Department of Computer Science, University of Bath

Abstract

This paper describes an approach for automatically producing convincing water surfaces from video data in real time. Fluids simulation has long been studied in the Computer Graphics literature, but the methods developed are expensive and require input from highly trained artists. In contrast our method is a low cost Computer Vision based solution which requires only a single video as a source. Our output consists of an animated mesh of the water surface captured together with surface velocities and texture maps from the video data. As an example of what can be done with this data, a modified form of video textures is used to create naturalistic infinite transition loops of the captured water surface. We demonstrate our approach over a wide range of inputs, including quiescent lakes, breaking sea waves, and waterfalls. All source video we use are taken from a third-party publicly available database.

Keywords: Video Processing, Water Surface Modelling, Water Surface Rendering

1 Introduction

Water simulation has been widely researched in the Computer Graphics literature. The major focus has traditionally been on better rendering quality and efficiency. Nowadays commercial tools for water modelling in various degrees of detail are used in a number of applications. These include photo-realistic visual effects production and less realistic but more economical effects for video games. However, it is typically the case that well trained artists are needed to use these tools. The primary motivation of the work we present here is to provide a low cost automatic method to acquire and display models of ordinary common-place bodies of water.

In this paper we address the problem of fluid acquisition and modelling using a fully automatic video based approach. The user only needs to shoot a video of a water surface using a single video camera for our system to approximate its geometry and its motion. The basic idea is that where water flow converges the surface must rise, whereas where the flow diverges the water falls. We show that a combination of shape from shading and incompressible flow lead to state of the art performance. Furthermore, the technique also works in real time. The output model is qualitatively very similar to the input video, which here we texture and loop infinitely — but other graphics applications are no doubt possible. Figure 1 gives an example of the input and output of our system.

In recent years, progress has been made in reconstructing complex objects and scenes from images or videos, for example: faces [5], human bodies [1] hair [13], trees [17] [16] and fluids [2]. Among them, water brings unique challenges, a solution to which is of great interest to many research areas within Computer Graphics [19]. Unfortunately, Computer Vision techniques are found to work less well for water than most other cases. This is due to several major challenges: a water surface generally lacks visually salient features; it exhibits complex photometric properties; it exhibits complex dynamics, including topological changes. These problems yield extreme difficulties for tracking, and ground truth data is difficult to acquire — even active acquisition systems such as laser scanners used by engineers fail due to the complicated reflection and refraction conditions.

This paper advances the current state of the art in image based water reconstruction to work with a single input video captured in ordinary outdoor conditions; more exactly where the water is a suspension of particles such as mud. Compared to the gamut of related work (Section 2), this makes our approach unique. Moreover since water as a suspension is common in the environment (as opposed, say, to perfectly clear water with a regular texture on a flat bottom) our approach is useful to ordinary users. The proposed method has the following characteristics:

- It is designed to work with a single input video recorded using an ordinary capture device. All of the example videos we show were recorded by a consumer-level digital video camera in an outdoor environment.
- It is practical, efficient and stable. It runs in realtime because no complex optimization schemes are used.
- Experiments also show it performs consistently well across different scenarios with fixed parameters.
- The output models can be used in many Graphics applications. For example, they can be re-textured, played forever, and viewed from different angles.

In the remainder of this paper we first overview relevant background work (Section 2) before moving on to explain our acquisition (Section 3) and provide examples of graphics applications (Section 4). We provide results in Section 5 and also in supplementary material, further work is discussed in Section 6. We conclude in Section 7 that our method

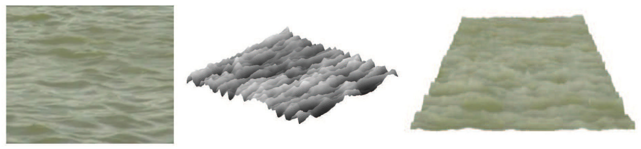

Figure 1: Our system takes a single input video (left) filmed by a consumer level camera and produces a height field of the water surface (middle) in realtime. The surface can be textured, lopped forever, and viewed from different view points in real time (right).

quickly produces realistic results at low cost to users, and has the potential to be yet more versatile.

2 Related Work

Reconstructing water surface is of interest to a wide range of groups: engineers, scientists and artists number amongst them. Since we have opted for a Computer Vision based approach to acquisition, this section confines itself to that area. Computer Vision has used various types of physical properties to reconstruct the water surface geometry, that depends on assumptions about the conditions in which video data was collected, as we now explain.

Refraction is one property that several researchers have exploited. Murase [11] reconstructs a water surface from the apparent motion of a refracted pattern. The distortion of an underwater pattern is tracked by optical flow, from which the water's surface normal is calculated using a refraction model. The water surface is then recovered by 2D integration of the surface normals (needle map). Balschbach *et al.* [3] also use a refraction approach, but based on a shape from shading technique where multiple illuminations are used to better determine surface gradients. Morris and Kutulakos [10] show that refractive index is not indispensable by assuming light is refracted only once. Their system reconstructs the water surface by minimizing the refractive disparity. These refraction based methods are generally called "shape from distortion" and they work well for transparent water. The disadvantages are they do not work with opaque liquids (more exactly, less transparent) and require specially designed devices to capture the distortion of a known pattern being located underneath the surface of the water. None of these methods are suitable for outdoor conditions, where water is a suspension and (to a first approximation) opaque.

Shape from stereo techniques have been explored to reconstruct liquids that are opaque. Wang *et al.* [19] dye water with white paint and light patterns are projected onto its surface. A depth field is first reconstructed by dense reconstruction and then refined using physically-based constraints. This method shows very accurate reconstructions of surface details. Ihrke *et al.* [8] dissolve the chemical Fluorescein in the water and measures the thickness of the water from the amplitude of the emitted light. The visual hull of the water surface is then calculated by utilizing weighted minimal surfaces using the thickness measurements as constraints. Hilsenstein [6] reconstructs water waves from thermographic image sequences acquired from a pair of infrared cameras. As a viable approach, infrared stereo reduces the problem associated with transparency, specular reflection and lack of texture at visible wavelengths. These techniques all require sophisticated equipment and complex experimental setups.

Missing from the literature is a solution for reconstructing water surfaces from a single video captured in an ordinary outdoor environment. This paper demonstrates that shape from shading combined with water incompressibility constrains optical flow, and performs consistently well across different types of such water surfaces.

3 Surface Acquisition

In this paper a height field $h(x, y, t)$ is used to represent the water surface at time t. Because this geometric representation assigns only a single height value to each (x, y) position, it rules out the realistic rendering of some effects such as convincing splashes, breaking waves, droplets or sprays. Nonetheless, we will show that a simple representation can still produce sufficient approximations, even for complex scenarios.

3.1 The Mass-Conservation Constraint

As stated above, the most important motivation of this work is to provide a low cost way to acquire models of commonplace bodies of water. Our solution tracks the water filmed by a consumer-level video camera and uses flow divergence to compute the change in height $\delta h(x, y, t)$ via the law of mass conservation. This section explains how.

Suppose the flow at each point on the water surface is known to be (u, v, w); later we will use optical flow to estimate it. The law of mass-conservation in the Navier-Stokes equations constrains the 3D divergence of the velocity to zero, which leads to

$$\frac{\partial w}{\partial z} = -\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right). \tag{1}$$

This links the (spatial) gradient in the vertical velocity component to the (spatial) gradient of the tangential velocity components. If these tangential velocities are invariant to height then $\partial w/\partial z$ is constant over z too. We set the boundary condition $(u, v, w) \cdot \mathbf{n} = 0$ where $n = (0, 0, 1)$ is normal to the water bed; hence $w = 0$ at $z = 0$. Integration over z now yields

$$w = h\frac{\partial w}{\partial z} = -h\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right). \tag{2}$$

The vertical velocity can can also be calculated from the material derivative of the surface height with respect to time:

$$w = \frac{dh}{dt} = \frac{\partial h}{\partial x}u + \frac{\partial h}{\partial y}v + \frac{\partial h}{\partial t} \tag{3}$$

Here we simplify the fluid dynamic by not considering the advection part $\frac{\partial h}{\partial x}u + \frac{\partial h}{\partial y}v$. Hence the Eulerian measurement of the surface change is used as an approximation of the vertical velocity $w \approx \frac{\partial h}{\partial t} = h(x, y, t+1) - h(x, y, t)$. The evolution of the water surface can then be directly linked to horizontal velocities via:

$$h(x, y, t+1) - h(x, y, t) = -h(x, y, t)(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}) \quad (4)$$

Estimates of the horizontal flow (the divergence) can therefore be used to estimate the change in height relative to the current height. Equally, estimates of the height change can be used to constrain horizontal velocities. The rest of this section first demonstrates how shape from shading can be used to acquire a prior for the water surface and then explains how to use such a prior to improve the tracking of horizontal velocities. The geometry and velocity of the final water surface model conform to the mass conservation constraint.

3.2 Surface Estimation

The Horn-Shunk algorithm is one the most widely used optical flow techniques [7]. They assume that the intensity of a point remains constant over a short time interval. If a point at location (x, y) at time t has intensity $I(x, y, t)$ and moves with velocity (u, v) then the energy function is given by

$$E(u, v) = \int\int (I_x u + I_y v + I_t)^2 + \alpha^2(||\nabla u||^2 + ||\nabla v||^2)dxdy \quad (5)$$

in which I_x, I_y, I_t are the first partial derivatives of I, and α a constant weighting factor. The velocity (u, v) that minimises this energy term is taken to be the optical flow. However, this method is not suitable for computing optical flow over water because the regularising term imposes a constraint that is not directly related to the physics of water. Others have noted this and placed physical constraints on flow. Doshi and Bors [4] use a robust kernel which adapts to the local data geometry in the diffusion stage of the Navier-Stokes formulation. The kernel ensures that smoothing occurs along the structure of the motion field while maintaining the general optical flow structure and the main optical flow features. Sakaino [14] proposes a method to model abrupt image flow change. Flow is modelled using a number of base waves and their coefficients are found to match the input sequence. Nakajima *et al.* [12] propose an energy function as a weighted combination of conservation of intensity, conservation of mass, and momentum equations.

All of the above offer some improvement over Horn and Shunk's general purpose formulation for the special case of water. Yet all of them tacitly assume that all motion is parallel to the viewing plane, that is all assume motion is confined to two dimension. This assumption is clearly breached by a moving water surface.

Our idea is to use shape from shading water surfaces as a prior to constrain the tracking of horizontal velocities based on the conservation of mass. We use the method described in [18] to acquire an initial estimate of height $h(x, y, t)$. Experiments also show shape from shading can work for dynamic water with very few adaptations (which surprised the authors of this paper, but is nonetheless true).

Videos are low-pass filtered to remove noise, such as extreme bright or dark points. Then for a video of length T frames and a resolution of M by N [1], the average height of each surface $\frac{1}{MN}\sum_{i=1}^{N}\sum_{j=1}^{M} h(i, j, t)$ is rectified to the same level $\frac{1}{TMN}\sum_{k=1}^{T}\sum_{i=1}^{N}\sum_{j=1}^{M} h(i, j, k)$ to remove the affection of global luminance change as

$$h'(x, y, t) = h(x, y, t) - \frac{1}{MN}\sum_{i=1}^{N}\sum_{j=1}^{M} h(i, j, t) + \frac{1}{TMN}\sum_{k=1}^{T}\sum_{i=1}^{N}\sum_{j=1}^{M} h(i, j, k). \quad (6)$$

Although shape from shading has limitations, it captures the surface geometry of different types of water and provides a good prior to constraint the horizontal velocity tracking.

Given the shape from shading surfaces, the vertical velocity w is approximated as their Eulerian derivatives with respect to time. Its gradient along the z-direction $\frac{\partial w}{\partial z}$ is consequently calculated as $\frac{h(x,y,t+1) - h(x,y,t)}{h(x,y,t)}$. The horizontal velocities (u, v) are then whatever it takes to make the water incompressible. The objective energy function is a weighted combination of intensity-conservation, mass-conservation and smoothness:

$$E = \int\int [(I_x u + I_y v + I_t)^2 + \alpha^2(|\nabla u|^2 + |\nabla v|^2) + \beta^2(u_x + v_y + w_z)^2]dxdy \quad (7)$$

$(I_x u + I_y v + I_t)^2$ and $|\nabla u|^2 + |\nabla v|^2$ are the intensity-conservation term and smoothness terms from the Horn-Schunck [7] optical flow. $(u_x + v_y + w_z)^2$ is the mass-conservation term that describes the 3D divergence of the velocity. In practice, w is calculated by subtracting the current shape from shading surface from its successor. Then w_z is calculated as w/h. The following Euler-Lagrangian equations are used to minimize the objective function 7:

$$I_x(I_x u + I_y v + I_t) - \alpha^2 \triangle u - \beta^2(u_{xx} + v_{xy} + w_{xz}) = 0 \quad (8)$$
$$I_y(I_x u + I_y v + I_t) - \alpha^2 \triangle v - \beta^2(u_{xy} + v_{yy} + w_{yz}) = 0 \quad (9)$$

In practice $\triangle u$, $\triangle v$, u_{xx} and v_{yy} are approximated numerically using finite differences. For example, the Laplacians are approximated as $\triangle u = \frac{1}{4}(u(x-1, y) + u(x+1, y) + u(x, y-1) + u(x, y+1)) - u(x, y)$ and $\triangle v = \frac{1}{4}(v(x-1, y) + v(x+1, y) + v(x, y-1) + v(x, y+1)) - v(x, y)$. The Lagrange multipliers α^2 and β^2 are fixed to 1000 across all scenes. The solutions of equations 8 and 9 are found using the Gauss Seidel

[1]All videos in this paper have resolution of 352 by 288.

method. The resulting horizontal velocity (u, v) is then used to calculate the final vertical velocity w and the change of the height water surface is re-estimated using equation 4. Since the system (Equations 8 and 9) is linear, it does not bring extra complexity than Horn and Shunck and can be solved in realtime.

3.3 Intermediate results

Figure 2 shows instantaneous surfaces computed using our mass-conserved flow. The results suggest we can recover a convincing surface even from dynamic water, such as breaking waves and water falls. Strong shadows, reflections and foreign bodies can produce artefacts — removing these is future work. However, the convincing results for the majority of cases we have tested demonstrate that simple shape from shading can be an effective constraint. In particular, our mesh moves in time whereas that computed by Horn-Shunk [7] is static, and that computed using Nakajima et al. [12] barely moves. Our mesh also includes an estimate of the instantaneous velocity (u, v, w) at each point. Comparisons with prior art and examples of velocity meshes are available in the supplementary material.

The next step is to show the mesh is of value to Computer Graphics. We do so in this paper by texture mapping the mesh, and creating naturalistic infinite transition loops of the captured water surface.

4 Example Graphics Applications

There is a potentially unlimited variety of graphics applications that require the rendering of water in some form or another, *e.g.* VFX shots and water surfaces in video games. In order to further the utility of the water capture method already presented, we now describe an approach to texture map the data and allow naturalistic infinite looping of the captured sequences. In order to achieve the latter, we modify the *Video Textures* algorithm [15] as explained below.

4.1 Texture Mapping

The general approach we take for texture mapping the captured water surfaces is to apply the video frames to the corresponding surface meshes. We construct a surface mesh from the input height field using triangles, and then determine what colour we will make the surface mesh. In our texture mapping application this corresponds to identifying texture coordinates in the corresponding video frame. For each point in the surface mesh we allocate a texture coordinate, which dictates where that particular pixel's colour comes from.

At this point, we backward map from points on the 3D surface to the texture using UV texture coordinates. Bilinear interpolation filtering is used to colour points on the surface which do not lie exactly on a texture coordinate. This achieves a better result when our texture is inspected closely. In addition, we use mip-mapping [20] to allow for rapid zoom-in and zoom-out.

4.2 Video Textures

In this section we describe our approach for making a water sequence loop infinitely in a naturalistic way. This depends on analysing the component frames of a video to produce the correct looping effect. Our approach is an extension of that originally presented by Schödl's [15]. As we explain, we cannot directly use this algorithm as the sequences we use are short and water is quasi-periodic (especially turbulent water).

The first step in our technique is to perform a pairwise comparison between each frame in the source video to compute a measure of similarity. Like Schödl, we chose to compare the pixel colours in the frames by using the Euclidean norm. Rather than compute this on each colour channel we chose to use the gray level representation of the images. The Euclidean norm d between frame a and frame b is defined as

$$d(a, b) = \left(\sum_{k=1}^{N} (a_k - b_k)^2 \right)^{1/2}, \tag{10}$$

in which k indexes a pixel. We use this to build up a matrix

$$D_{ij} = ||I_i - I_j|| = d(I_i, I_j), \tag{11}$$

which contains all the computed similarities between each pair of frames in the input video, where i and j range over all of the frames in the input video. We normalise D such that the sum of values is unit, that is $\sum_{ij} D_{ij} = 1$, as this makes computations more convenient later.

The next step is to consider the probabilities of transition from one frame to the next. Normally this would be the successive frame in the sequence. However, in order to create a convincing infinite sequence from a finite video we must consider transitions to other frames. If we are at frame i, we may wish to transition to some frame j whenever the distance, D, between $i + 1$ and j is small. Following Schödl [15], we assume the transition probability is Gaussian distributed with D the (squared) Mahalanobis distance, thus

$$P_{ij} = exp\left(\frac{-D_{i+1,j}}{\sigma} \right) \tag{12}$$

is the transition probability from from frame i to frame j. We use this to make the choice to the next transition. Here, σ is a scalar which controls the degree to which similar frames can be jumped to: a small value means only very similar frames are likely, and large value yields more-or-less random jumps. We set $\sigma = 1.2 \times 10^{-4}$ for all video sequences.

The matrix P_{ij} is a zeroth-order Markov chain and therefore ignores all dynamic data. As Schödl [15] point out this can have a significant impact. For example, suppose we wish to create a video texture from a pendulum, swinging back and forth. There will be a point as the pendulum ascends which is kinematically identical to when it descends, but dynamically its mirror. Transitioning at this point can produce a strange animation where the pendulum's arm appears to abruptly change direction without cause.

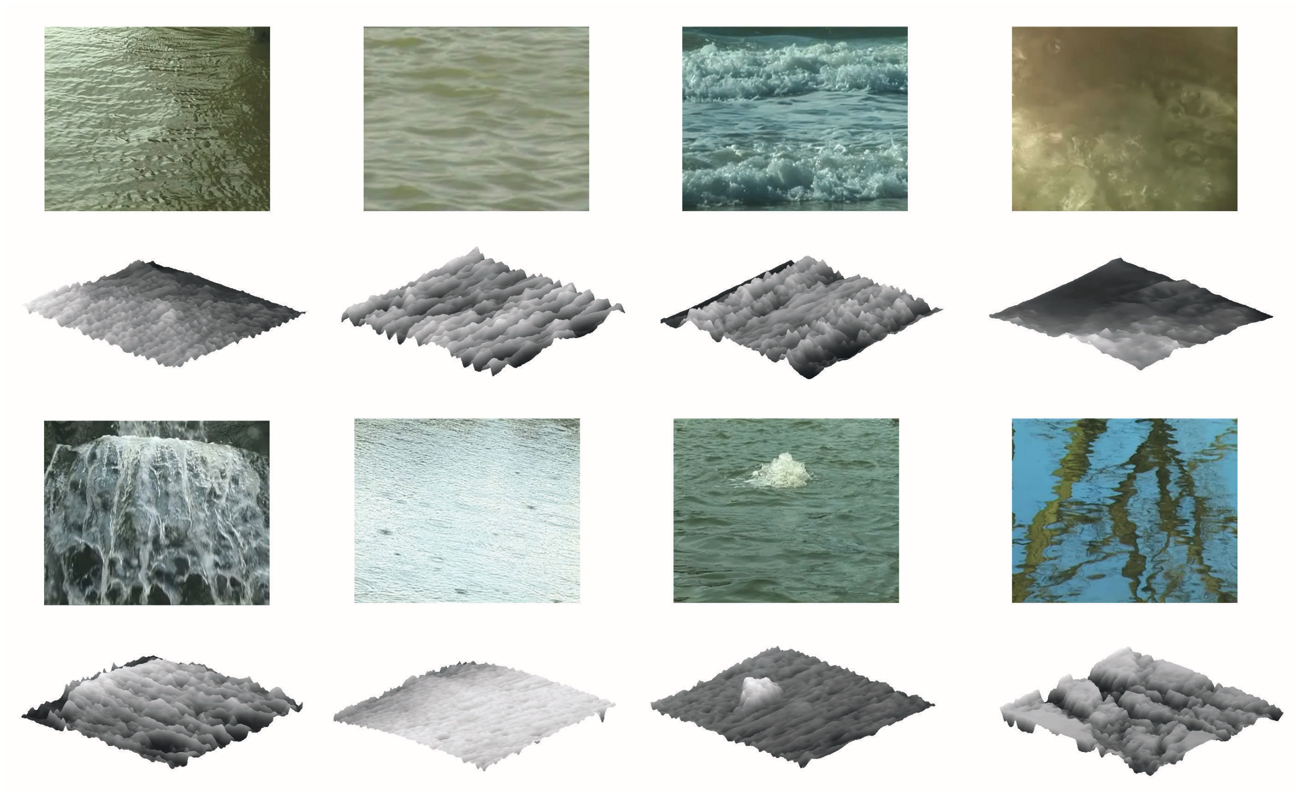

Figure 2: Ten example videos and their shape from shading surface reconstructions. The reconstructed surfaces are rendered from a different view point to reveal the 3D information. Top row, left to right: 'breezey ripples', 'quiescent', 'waves', 'boiling'; bottom row left to right: 'water fall', 'rain', 'fountain', 'reflection'. Notice the 'reflection' surface is an example of the failure cases.

Our algorithm produces optical flow vectors, which could be used to preserve dynamics. Schödl *et al.* advise against this however, stating that flow computations can be quite brittle. They suggest a simpler alternative – in order for one frame to be classified as similar to another frame temporally adjacent frames within a weighted window must also be similar. We follow their advice and match a subsequence of frames rather than individual frames. We compute this, following Schödl *et al.* [15], as follows:

$$D' = \sum_{k=-m}^{m-1} w_k D_{i+k,j+k} \tag{13}$$

We use $m = 2$ in our implementation, which corresponds to a 4 tap filter, and we set w to be binomial coefficients. We can then convert D' to probabilities, as before.

Knowing that frame i is similar to frame j is not enough for video textures — consider the pendulum example which was discussed previously. Again following Schödl *et al.* we reduce the probability of a sudden reversal in the direction an object moves by matching frames that neighbour both i and j, in effect matching a mini-sequence of 3 or so frames. However, even this is not sufficient in our case: we found that standard video textures always produces a visible discontinuity in the clips we used — the supplementary material contains some examples.

It is true that Schödl *et al.* provide examples of video texture using water, so some explanation is required as to why it fails in our cases. Our explanation is this: we use short video clips that are close to the water surface and we reconstruct three dimensional surfaces, these conditions act to highlight visual discontinuities. Schödl *et al.*, on the other hand, use clips of much longer length and in which the water is viewed from a greater distance, they make no effort to recover three dimensional information; all of which act to suppress visual glitches. Kwatra [9] point to the same problem as we do (although they do not reconstruct surfaces). They solve the problem using graph cuts, we have developed a much simpler technique but one which nonetheless produces acceptable visual results.

4.3 Video Texture transitions for water

There are two ways our work deviates from standard video textures, both of which are rooted in the chaotic nature of water, including the relatively calm examples we have. Consider the current frame to be frame i and the target frame to be j. The two departures are:

1. we re-shape the transition probability P_{ij} using a shaping function so that j is not close to i; and
2. we transit from i to frame j using a sliding interpolation akin to cross-fade.

The rationale behind the first of these is that the transition probabilities P_{ij} decay very rapidly as frame j moves away in time from frame i. At a sufficient distance the probabilities do exhibit oscillations as expected, but even the peak values are much lower than the probabilities close to frame i. This has an important effect: if we were to pick the target frame j under the distribution P_{ij} as defined in Equation (even if D' replaces D), then we tend to end up cycling over just a few frames of video. To solve this we could increase σ, but that leads to a nearly flat distribution which is not wanted; we therefore re-shape P_{ij}. This leads directly to the rationale behind the second departure: the target frame j from frame i differs so much from i that a simple transition as advocated by Schödl *et al.* results in the visible glitches mentioned above. We discuss the shaping function first.

Our goal is to find a pair of frames in the video which can provide us with a subtle transition from frame i to frame j. However, as explained frame j should be distant from i if we are not to loop over just a few frames. We choose a shaping function that rises away from zero as frame j become more temporally distant from i and remains at unity once j is sufficiently distant. Any sigmoid function would do for this, we chose the error function which is defined by

$$erf(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt. \tag{14}$$

In practice, the shaping function we use, $q(j, i)$, has a 'flat bottom', and is defined by

$$q_{j,i} = \begin{cases} 0 \text{ if } j \in [i-\mu, i+\mu] \\ erf(|v_1|\lambda) \text{ if } j > i+\mu \\ erf(|v_2|\lambda) \text{ if } j > i-\mu \end{cases} \tag{15}$$

in which $v_1 = j - (i + \mu)$ and $v_2 = j - (i - \mu)$ are the distances of frame j from the upper and lower bounds of the closed interval $[i - \mu, i + \mu]$ in which the transition probability is clamped to zero. The λ is a constant that controls the rate at which the function climbs from zero to unity. We set $\lambda = 1/8$, and μ equivalent to two seconds of video — so $\mu = 48$ at 24 frames per second. We then re-compute the transition matrix so that

$$P'_{ij} = P_{ij} q_{ij}. \tag{16}$$

We now address the problem of transitioning between one part of the video and another. First we match frame $i + 1$ to some frame j, using the transition matrix in 16. Were we to follow standard video textures we would suppose the current frame is i and its the successor, frame $i + 1$, to some frame j; then continue the video from frame j. However, as mentioned above, standard video textures does not work with our video clips, because even relatively still water is too chaotic so there is always a noticeable discontinuity. We therefore take a temporal window of width K and blend the sequence $I(i - 1 - K)$ to $I(i + 1 + K)$ with the sequence $I(j - K)$ to $I(j + K)$. The blending function is a linear slide over the sequences using a pseudo-index k; the k^{th} output frame is given by

$$I'(k) = (1 - w(k))I(i + 1 + k) + w(k)I(j + k). \tag{17}$$

in which $w(k) = (k + K)/2K$, for $k = -K, -K + 1, ..., K$. Examples showing the result of this approach may be found in the supplementary material.

Since we are now using a blended sequence rather than replacing frames we must ensure the jump does not require a sequence that exceeds the video boundary. For example, jumping to frame one would not be allowed because that would require us to access frames before frame one. We overcomes this by clamping the shaping function to 0 at the start and end of the video.

5 Results and Discussion

We show the results of our technique using a number of dynamic textures from the DynTex database [2]. We partition our results into surface acquisition and surface rendering. Each of these is further partitioned into failure cases of prior art, our success cases, and our failure cases that point to future work.

Our method only uses a linear optimization for model fitting (Equations 8 and 9) so there is no significant increase of computational complexity when compared to the traditional optical flow algorithms such as [7]. All the outdoor videos used in this paper have a resolution of 352 by 288. For such a resolution, the shape from shading algorithm [21] is also able to run in realtime for the initial surface acquisition. Hence overall our method is able to produce the output model in realtime. There is a small overhead for computing the similarity matrix for video texture. However, this only needs to be done once for each video. Once the similarity matrix has been computed, the rendering is in realtime, as shown in the demo videos.

Ideally the captured model should be quantitatively evaluated. In practice we realized it is difficult to get the ground truth data for water surfaces. Even active acquisition systems such as laser scanners will fail due to the over complicated reflection and refraction conditions. Authors of [19] project random patterns to water that has been dyed with white paint and use stereo reconstruction to capture its surface in an indoor environment. So far as we know this is arguably the best available ground truth approximation for water surface. However, it is not possible to use this method for outdoor scenes as large scale water can not be dyed. Hence we leave the ground truth water surface capture as an interesting future avenue and focus on the qualitative evaluation in this paper.

Acquisition results have already been briefly discussed in

[2]http://projects.cwi.nl/dyntex/

Subsection 3.3. As mentioned there, supplementary material provides evidence that current optical flow algoirthms, including those designed specifically for water, fail to produce convincing moving surfaces. It also shows we produce flow vectors that appear by qualitative inspection to be of higher quality than those produced by prior art. Returning to Figure 2 we see that the shape from shading constraint is useful in many cases, but fails where there a strong cast shadow, a foreign body who's intensity colour contrasts greatly with the surrounding water, or (as shown) reflections. Overcoming this limitation is future work.

Figure 3 shows single frames from our rendering results. The 'flotsam' example shows that shape from shading can be robust to foreign objects, provides they offer little illumination contrast with the water surface. The 'froth' example shows a highly reflective surface in which the optical properties are far from the diffuse assumptions made for shape from shading. In this case the surface obtained in not entirely veridical but is qualitatively convincing. Figure 4 shows examples of failure cases. Example 'break' highlights an area showing a peaking wave, just about to break — however in the surface we see a trough. We conclude the best surfaces tend to be produced when the perspective of the camera is almost directly above the water surface, which is in line with our assumptions linking vertical velocity to tangential velocity. The 'duck' example shows that a dark foreign body leads to a trough in the surface; this is more obvious with the 'shadow' example. The supplementary material contains the texture mapped video all of which are rendered as an infinite loop via video textures. It shows the visible discontinuities using standard video textures that motivated our modification, and video of our successes corresponding to Figure 3, further successes and failures cases (which, again, can be traced back to our use of shape from shading).

6 Further Work

There are several future directions for this work. The shape from shading issue is one obvious avenue to make acquisition operate more robustly over a wide gamut of inputs. We may also attend to details of the surface fitting procedure so that sharper features are better preserved.

In terms of rendering, this paper has demonstrated feasibility but there is much more that could be done. For example, there is the issue of scene lighting from alternate perspectives. This is a potentially costly computation that our technique is designed to avoid. However, given our results indicate that the surface appears natural unless the view is varied too far from the original source — an observation that might be used to build a fast solution.

Secondly, we could improve the visual quality of our rendered output with the use of particle systems. We have observed that when water gets sufficiently turbulent, we see entities of water shoot through the air. This is most noticeable in breaking wave situations. We could look for these instances through optical flow vectors computed from the source video. We could then fire particles to emulate this behaviour in our water surfaces.

Our technique only considers small, rectangular patches of water. In many applications, we desire the shape and size of our water surface to be arbitrary. Texture synthesis techniques [9] would allow us to enlarge the size of our source video and hence we could produce such water surfaces.

Although our method for interpolating between transitions in video textures produces acceptable results on a wide range of test cases, the graph cut technique employed by Kwatra *et al.* [9] has been shown to be more robust at the cost of more complexity. An extension to this project could be to implement a graph cut technique like this to transition between frames, in place of our interpolation method. However, this would not help us in the challenging cases when attempting to produce a video loop within a breaking wave situation. Kwatra *et al.* show an example of their technique on such a case in their demonstration video[3] and find results similar to those which our technique achieved. Unfortunately, to our knowledge, there is no solution to seamlessly loop videos in these situations.

Finally, we emphasise our graphics applications are intended to be examples of what is possible rather than definitive of what can be done. We can imagine applications based on ray tracing, particle systems, and even non-photorealistic rendering in cases where a cartoon like feel is important (for example).

7 Conclusion

This paper studied the problem of video-based water reconstruction for a single input video that is captured in ordinary outdoor conditions. In this case the prior art based on refraction and (stereo) reflection based reconstruction techniques are impractical. We have demonstrated the robustness of the method using many different types of water. The advantages of the proposed method are: 1) it works fully automatically and requires only basic input resources; 2) it is very efficient and can run in realtime; 3) The model can be textured, lighted and looped for interesting graphics applications. Furthermore, our approach requires no specialist hardware: we are able to produce these water surface animations from standard quality digital cameras. As expected with any depth estimation technique, the quality of the output increases with the resolution and lens quality of the camera.

One important discovery is the capability of shape from shading as a constraint to recover different water surfaces of this kind. Consistent performance is demonstrated by experimenting on a wide range of scenes.

There are many interesting future avenues for this work, both in surface acquisition and in rendering.

[3]http://www.cc.gatech.edu/cpl/projects/graphcuttextures/

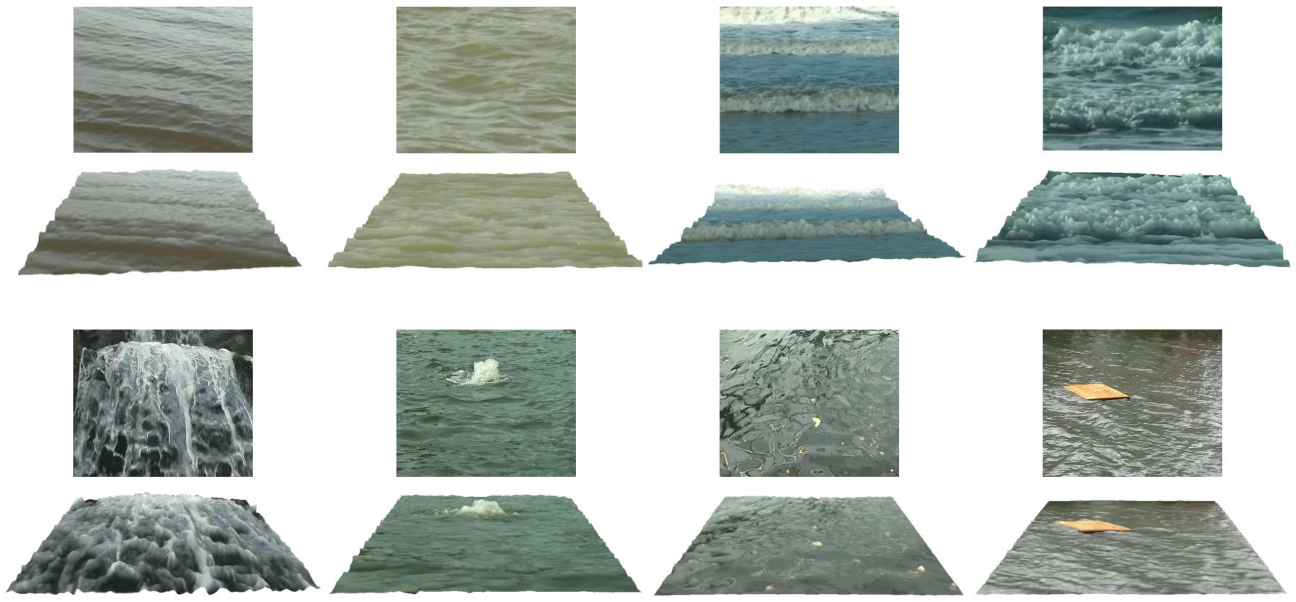

Figure 3: Images from several challenging water sequences along with their corresponding reconstructed texture mapped surfaces. Top row, left to right: 'breezy ripples', 'quiescent', 'waves', 'waves too'. Bottom row, left to right: 'water fall', 'fountain', 'flotsam', 'froth'.

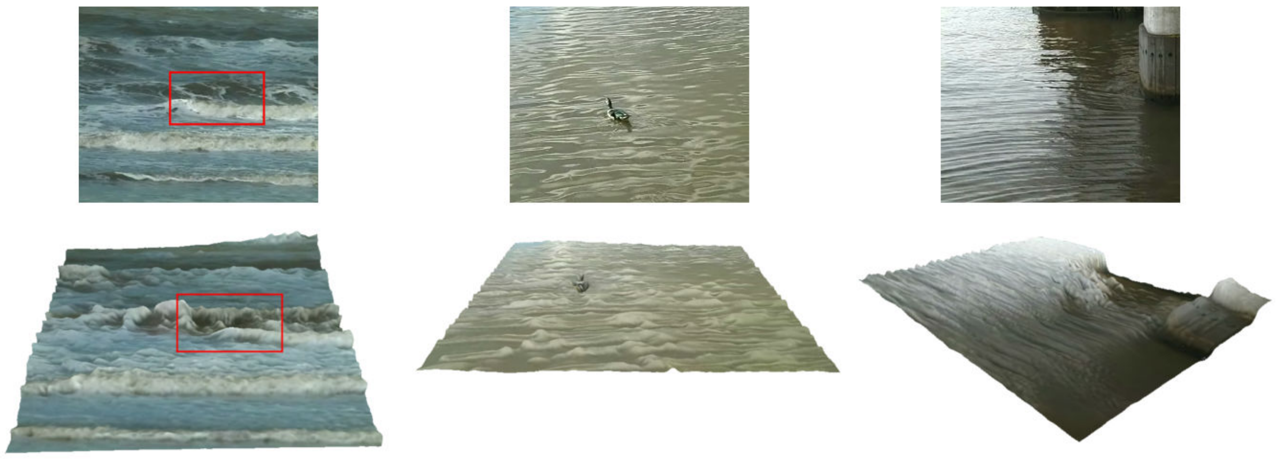

Figure 4: Several pairs of video frames and their corresponding surface meshes demonstrating failure cases. Left to right 'break', 'duck', 'shadow'. See Figure 2 for a reflection example (not texture mapped).

References

[1] E. D. Aguiar, C. Stoll, C. Theobalt, N. Ahmed, H. Seidel, and S. Thrun. Performance capture from sparse multi-view video. *Proceedings of ACM SIGGRAPH*, 27(3):1–10, 2008.

[2] B. Atcheson, I. Ihrke, W. Heidrich, A. Tevs, D. Bradley, M. Magnor, and H. Seidel. Time-resolved 3d capture of non-stationary gas flows. *Proceedings of ACM SIGGRAPH Asia*, 27(5):1–9, 2008.

[3] G. Balschbach, J. Klinke, and B. Jähne. Multichannel shape from shading techniques for moving specular surfaces. In *Proceedings of the European Conference on Computer Vision*, pages 170–184, 1998.

[4] A. Doshi and A.G. Bors. Navier-stokes formulation for modelling turbulent optical ow. In *British Machine Vision Conference*, page 110, 2007.

[5] A. Ghosh, T. Hawkins, P. Peers, S. Frederiksen, and P. Debevec. Practical modeling and acquisition of layered facial reflectance. *Proceedings of ACM SIGGRAPH Asia*, 27(5):1–10, 2008.

[6] V. Hilsenstein. Surface reconstruction of water waves using thermographic stereo imaging. In *Image and Vision Computing New Zealand*, pages 102–107, 2005.

[7] B. K. P. Horn and B. G. Schunck. Determing optical flow. In *Artificial Intelligence*, volume 17, pages 185–203, 1981.

[8] I. Ihrke, B. Goldluecke, and M. Magnor. Reconstructing the geometry of flowing water. In *Proceedings of the International Conference on Computer Vision*, pages 1055–1060, 2005.

[9] Vivek Kwatra, Arno Schödl, Irfan Essa, Greg Turk, and Aaron Bobick. Graphcut textures: image and video synthesis using graph cuts. *ACM Trans. Graph.*, 22:277–286, July 2003.

[10] N. J. Morris and K. N. Kutulakos. Dynamic refraction stereo. In *Proceedings of the International Conference on Computer Vision*, pages 1573–1580, 2005.

[11] H. Murase. Surface shape reconstruction of a nonrigid transport object using refraction and motion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(10):1045–1052, 1992.

[12] Y. Nakajima, H. Inomata, H. Nogawa, Y. Sato, S. Tamura, K. Okazaki, and S. Torii. Physics-based flow estimation of fluids. In *Pattern Recgonition*, volume 36, pages 1203–1212, May 2003.

[13] S. Paris, W. Chang, O. I. Kozhushnyan., W. Jarosz, W. Matusik, M. Zwicker, and F. Durand. Hair photobooth: geometric and photometric acquisition of real hairstyles. In *Proceedings of ACM SIGGRAPH*, pages 1–9, New York, NY, USA, 2008. ACM.

[14] H. Sakaino. Motion estimation method based on physical properties of waves. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2008.

[15] Arno Schödl, Richard Szeliski, David H. Salesin, and Irfan Essa. Video textures. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '00, pages 489–498, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.

[16] P. Tan, T. Fang, J. X. Xiao, P. Zhao, and L. Quan. Single image tree modeling. *Proceedings of ACM SIGGRAPH Asia*, 27(5):1–7, 2008.

[17] P. Tan, G. Zeng, J. D. Wang, S. B. Kang, and L. Quan. Image-based tree modeling. In *Proceedings of ACM SIGGRAPH*, page 87, New York, NY, USA, 2007. ACM.

[18] P. Tsai and M. Shah. Shape from shading using linear approximation. *Image and Vision Computing*, 12:487–498, 1994.

[19] H. M. Wang, M. Liao, Q. Zhang, R. G. Yang, and G. Turk. Physically guided liquid surface modeling from videos. In *Proceedings of ACM SIGGRAPH*, pages 1–11, 2009.

[20] Lance Williams. Pyramidal parametrics. *SIGGRAPH Comput. Graph.*, 17:1–11, July 1983.

[21] R. Zhang, P. Tsai, J. E. Cryer, and M. Shah. Shape from shading: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21:690–706, 1999.