Eikonal Approximation for Real-Time Height Field Caustics

IFT6042 - Final Report

Olivier Mercier*
Université de Montréal

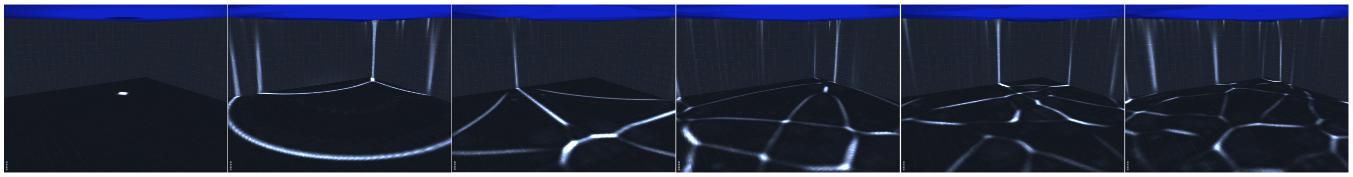


Figure 1: *Evolution of caustics in the* pool *scene. The water surface is initialized as a Gaussian bump and animated with a wave simulation.*

Abstract

Caustics are a key ingredient to produce believable refractive fluid simulations. However, their complex nature makes them very costly to evaluate accurately. For this reason, caustics are often omitted in real-time applications. In this paper, we present a novel method for creating caustics using the eikonal equation. We focus mainly on underwater caustics created by a water interface represented by a height field, with emphasis on interactive frame rates.

Keywords: caustic, height field, eikonal equation, real time.

1 Introduction

Modelling the behaviour of light is of prime importance for the realistic rendering of any scene. As light rays react to the objects they encounter, their density changes, which in effect modulates the amount of energy that reaches different parts of the scene. One obvious resulting effect is the formation of shadows in regions that receive less light. Caustics are in some sense the opposite of shadows and appear when light rays converge to a region that receives a greater amount of energy than its neighbouring region.

One important class of caustics is those created by a small reflective or refractive object. For instance, a transparent glass sphere placed between a diffuse surface and a light source will bend light rays and create a caustic pattern on the receiving surface. Those patterns are often bounded in space and can thus be simulated using localized techniques. Other caustics are caused by objects whose size is of the order of the entire scene. A usual example occurs when simulating an underwater scene in shallow water, where caustics are caused by the entire water surface. This surface can cover a large part of the scene, and its caustics will be found throughout the whole scene. Techniques for evaluating caustics in such environment thus have to be very efficient, especially for real-time applications.

*e-mail:mercieol@iro.umontreal.ca

2 Previous Work

Since caustics are created by the complex interaction of light with the surrounding scene, computing them exactly is a difficult task. To bypass this issue, early attempts have used precomputed texture stitching [Stam 1997] to simulate caustic effects. Another procedural approach was used by Liao et al. [2011] for applications to cartoon animations. Even if those methods do not give realistic effects, the patterns they create are plausible. This suggests that a physically based simulation is not necessarily required, as long as the general appearance of caustics is correct.

One of the first methods for creating physically based caustics uses backward ray tracing [Arvo 1986]. This intuitive method computes the illumination at an intersection point between a view ray and a diffuse surface by sending rays from this point towards the scene in hope of reaching light sources. This would in theory give perfect results, but the method suffers from all the usual problems of global illumination. More precisely, using too few light direction samples produces a high level of noise in the resulting images. Computing caustics with this approach in complex scenes where multiple refractive and reflective objects are present is therefore inefficient. Nonetheless, interactive frame rates can be achieved for simple scenarios such as the one of interest in this paper, e.g., caustics caused by a water interface represented by a height field [Yuksel and Keyser 2009].

Instead of computing caustics from the receiving surface to the light, direct methods can be used. Photon mapping [Jensen 2001] is one of the most used techniques of this kind. Light particles are emitted from the light sources and accumulated on the different surfaces they encounter. Illumination can then be computed by looking at the density of photons on each surface. This global illumination technique can be used to compute caustic patterns in a physically correct way [Jensen 1996]. Even if a very large quantity of photons usually have to be used, approximations and parallelization can be applied to the method to create caustics at interactive frame rates [Shah et al. 2007].

Beam tracing [Heckbert and Hanrahan 1984] is another direct approach where triangular beams of light are propagated in the scene instead of point particles. Those beams intersect surfaces at triangular regions which can be used to define caustics [Watt 1990]. The beams are easily created from the triangular mesh of the refractive object. The main challenge of this approach is to compute the illumination contribution from a beam of light intersecting a diffuse surface. One solution is to compute the intersection of beams of light with a scanning plane to accumulate the light contributions in image space [Nishita and Nakamae 1994]. Many other variations and improvements of beam tracing have been developed. For instance, Iwasaki et al. [2002] improve Nishita's method and optimize it for graphics hardware. In another variant, Brière and Poulin [2001] use beam tracing with hierarchical structures to accelerate render times without sacrificing accuracy. Beam tracing is still used in recent applications [Liktor and Dachsbacher 2011] where beams of adaptive sizes are used to obtain detailed results at affordable frame rates.

Even if those three classes of methods are able to achieve interactive frame rates, they can often only do so for simple scenes. Also, the algorithms are usually inefficient in their simplest version and only become usable through an optimized and complicated implementation. The goal of this paper is to study a different approach that is both efficient and easy to implement.

The field of computational physics has produced methods for computing wave propagation based on the eikonal equation. This differential equation models the propagation of a wave front by recording the minimal time at which it reaches each point in space. Solving this equation has given rise to powerful methods in geophysics, for instance for the simulation of seismic waves [Buske and Kästner 2004], and very similar methods can be applied to compute light propagation. The eikonal equation has already been used to accelerate ray tracing algorithms [Benamou 1996]. Similarly, Ihrke et al. [2007] precompute the eikonal solution and use it to determine efficiently the direction of incident rays. Their method also stores irradiance values at each node of the grid as the light travels through the scene and uses those values to integrate the irradiance along a viewing ray for participating media. Even though this approach is efficient, the precomputations require time and memory that is not available in real-time applications. Section 3 will study new ideas for using the eikonal equation in a more direct and reusable way.

3 Eikonal Interpretation

Light travels at different speeds in different media. Let $s(\mathbf{x})$ be the (scalar) speed of light at a point $\mathbf{x} \in \mathbb{R}^3$. We define the *propagation cost of light* at a point $\mathbf{x}$ by $F(\mathbf{x}) := \frac{1}{s(\mathbf{x})}$. Given a closed orientable surface S_0, the eikonal equation

$$\|\nabla\phi(\mathbf{x})\| = F(\mathbf{x}) \tag{1}$$

with boundary condition

$$\{\mathbf{x} \in \mathbb{R}^3 | \phi(\mathbf{x}) = \mathbf{0}\} = S_0 \tag{2}$$

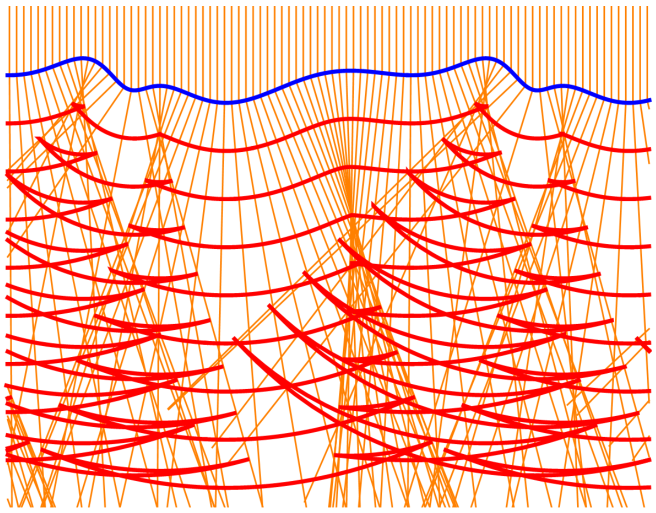
is a hyperbolic non-linear equation that describes the propagation of a wave front S_0 in a medium of propagation cost $F(\mathbf{x})$.

The solution ϕ to the eikonal equation, which we will denote from now on as the *eikonal solution*, gives the time at which a wave front reaches a given point in space. Therefore, the isosurface $\{\mathbf{x}|\phi(\mathbf{x}) = \mathbf{T}\}$ of the eikonal solution gives the position of the wave front at time T.

A widely used method for solving the eikonal equation is the Fast Marching Method (FMM) [Sethian 1996]. This method is based on the wave front interpretation of equation (1) and computes the eikonal solution on a grid. It is a continuous analogue to Dijkstra's algorithm for discrete graphs [Dijkstra 1959]. The method begins by initializing a set of grid nodes that will represent the initial wave front S_0. These nodes are flagged as *alive*. Then, the method searches for the grid node that is closest to the set of alive points with respect to the propagation costs F. It stores the travel time on this node and marks it as alive before looking for the next closest point. This process continues until all grid nodes are alive.

Under the assumption that light rays cross the interface perpendicularly (i.e., the receiving medium has an infinite refractive index), we can use the eikonal solution to compute the position of the wave front at a given time T. Figure 2a shows some of those wave fronts at different times. Due to the Hyugens principle, a wave front travelling in a homogeneous media (e.g. water) will expand orthogonally to itself. The wave fronts are thus always parallel curves. However, for the very common case of light rays going from air to water, the refractive indices are 1 for the air and 1.33 for the water. For such situations, the eikonal solution will be an incorrect approximation of the light front. Nevertheless, it is possible to modify the eikonal solution by scaling it in the average direction of refracted light rays. Figure 2b shows that this can give curves that closely fit the light front.

In this paper, we will use the infinite refractive index approximation so that we can use the eikonal solution. We leave it to future work to study how to modify our method for finite refractive indices.


(a) refractive index = ∞

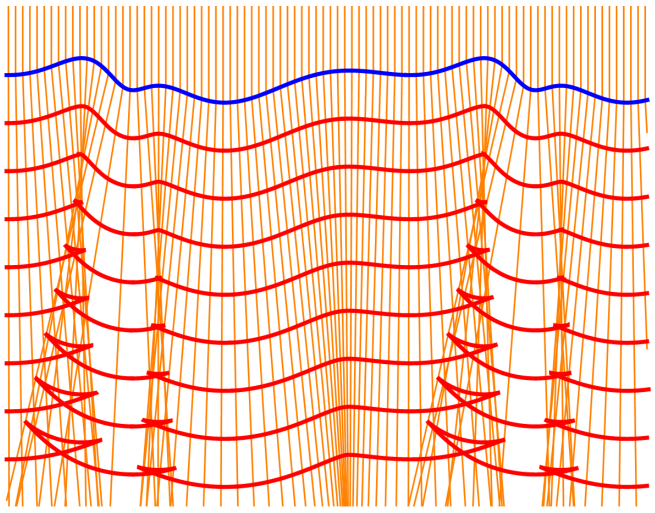

(b) refractive index = 1.33

Figure 2: *2D example of wave fronts (red) from an initial water-air interface (blue) at different times. Light rays (orange) are refracted from the interface. (a) Water has refractive index ∞ and the wave fronts are parallel curves to the initial interface. (b) Water has refractive index 1.33 and the parallel curves need to be modified in order to approximate wave fronts.*

If we restrict ourselves to the case of a water height field refracting light towards a planar sea floor, the FMM is very efficient. Since the light propagates from top to bottom, the search for the next closest point is facilitated. Also, once the wave front has entered the water, the propagation cost F is constant. We will assume $F = 1$ for simplicity.

Note that in figure 2, the wave fronts are computed using parametrized parallel curves, while the light rays are computed by simple refraction. We see that even starting with a smooth S_0, parallel curves can become singular over time, and those singularities are created where the light rays converge. This intuition is key to our method since it gives a way to detect regions in which caustics should be visible.

The following subsections discuss in more details some aspects of our eikonal approach.

3.1 Implementation with a 3D Texture

We implemented the basic Fast Marching Method on a regular 3D grid. The FMM is summarized in algorithm 1. The water-air interface is represented by a height field mesh. More details about the method can be found in the original paper [Sethian 1996]. We use a heap to store the *trial* cell, which makes the search for the next cell to update very efficient.

```
Initialize 3D grid with values $\phi = 0$;
for all vertices $v$ of the water-air interface do
    Find the cell $C$ in which $v$ falls;
    Set $\phi(c) \leftarrow \min(\phi(c), \text{distance}(v, \text{center of } c))$;
    Mark $c$ as accepted;
    for all neighbours $c'$ of $c$ not yet accepted do
        Mark $c'$ as a trial cell;
        Update $\phi(c')$;
    end
end
while not all cells are accepted do
    Find the cell $c$ with the smallest $\phi$ not yet accepted;
    Mark $c$ as accepted;
    for all neighbours $c'$ of $c$ not yet accepted do
        Mark $c'$ as a trial cell;
        Update $\phi(c')$;
    end
end
```

Algorithm 1: Fast Marching algorithm.

Once all cells are marked as *accepted*, the FMM stops. The grid then contains the values of the eikonal solution. In order to find the regions that are near singularities, we compute the curvature at each point of the grid, based on the intuition of section 3 that regions of high curvature correspond to regions around singularities. For a level set function ϕ such as the eikonal solution, curvature κ is computed as

$$\kappa(\phi) = \nabla \cdot \frac{\nabla\phi}{||\nabla\phi||} = \nabla \cdot \frac{\nabla\phi}{F} = \Delta\phi \tag{3}$$

and the Laplacian is evaluated with the simple discretization

$$\begin{aligned}\Delta\phi(x,y,z) &= \phi(x+1,y,z) + \phi(x-1,y,z) \\ &\quad + \phi(x,y+1,z) + \phi(x,y-1,z) \\ &\quad + \phi(x,y,z+1) + \phi(x,y,z-1) - 6\phi(x,y,z).\end{aligned} \tag{4}$$

The values are then transferred directly into a 3D texture. We will denote this texture as the *3D caustic texture*. Once this texture is computed, is can be used to project caustics on any object in the texture volume. This is the main advantage of our approach, since we can modify the objects in our scene and still compute caustics without any recomputation. Also, the caustics can be cast on any triangulated object without additional difficulty.

3.2 Boundary Conditions

The Fast Marching approach is easy to implement in the interior of the domain, but boundary conditions have to be treated carefully. Figure 3a shows a 2D case where the initial interface is linear. In this case, no caustics should be observed. However, the left-most cell will propagate in eccentric circles while the right part of the domain will propagate along lines. The junction between circles and lines (shown in red in figure 3a) will delimit a region of zero curvature and a region of positive curvature, which will create an undesired brighter region. This error is due to the fact that the left part of the plane that is outside of the domain cannot contribute to the eikonal solution.

There are two ways to solve this issue. The first is simply to embed the region of interest in a large enough texture domain, so that those boundary artefacts do not influence the scene. This solution can be very costly if the boundary errors propagate rapidly in the scene, since we need to have a large texture volume that is mostly unused.

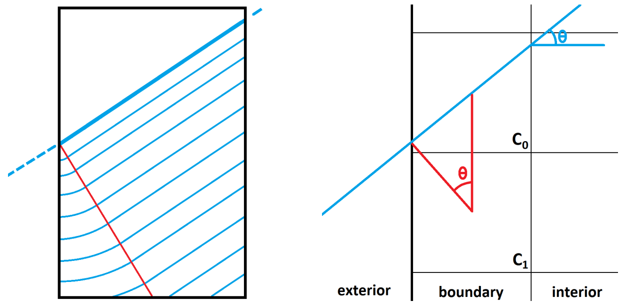
A more complicated but more efficient solution is to impose particular boundary conditions on the wave propagation by modifying the propagation costs at the boundary cells. Figure 3b enlarges the problematic region in figure 3a. Suppose we extend linearly the water-air interface at the boundary, and let C_0 bell the grid cell where the interface crosses the boundary. Let θ be the angle between the interface and the horizontal. The angle between the vertical and the normal to the extended boundary is also θ. If C_0 was to update the cell C_1 right below it, the result would be incorrect because the extended interface is supposed to reach C_1 α times faster than C_0, where $\alpha = \frac{1}{cos\theta}$. This coefficient α can also be computed as the inverse of the x component of the normalized tangent at the boundary. If we change the propagation cost at C_0 to be α, C_0 will reach C_1 at the correct time, and the eikonal value at C_1 will be correct. We apply this modification iteratively on all boundary cells, and then proceed to compute the FMM in the interior of the domain, thus eliminating boundary artefacts.

In 3D, the solution is fundamentally the same, but the implementation is more complicated. Each of the six boundaries of the cubic volume is treated independently. To solve the interior of each 2D boundary, we begin by initializing the cells that touch the water-air interface. On these cells, we will use the x and y tangents to modify the propagation cost as we did for the 2D case described previously. Figure 4a shows the modified propagation costs F at the interface on a 2D boundary. Doing a simple 2D propagation on this boundary using the initial travel costs is not sufficient, as regions with a low travel cost will propagate too quickly in all directions. We instead use a two pass approach. First, we propagate the boundary using $F = 1$ everywhere and record at each cell the F value of the cell that reached it. At the end of this first pass, each cell on the 2D boundary has been assigned a propagation cost, as shown in figure 4a. We then do a second pass using those F values to extend the boundary, as shown in figure 4b. The boundary treatment is summarized in algorithm 2. This method nearly eliminated the boundary artefacts in our results.

3.3 Loss of Information from the Viscosity Solution

As shown in figure 2, a point in space can be reached by different parts of the wave front at different times, which makes the

```
for each of the volume's six 2D boundaries do
    for each of the 2D boundary's four 1D boundary do
        if at least one cell touches the water-air interface then
            for each cell that touches the water-air interface do
                Compute $F$ from the water-air interface tangent;
            end
            Propagate the eikonal solution in the 1D boundary
            until all cells have been set;
        end
    end
    if at least one cell touches the water-air interface then
        for each cell crossing the water-air interface do
            Compute $F$ from the water-air interface tangent;
        end
        Propagate the F values in the 2D boundary at propagation
        cost 1 until all cells of the 2D boundary have been visited;
        Propagate the eikonal solution in the 2D boundary until
        all cells are set;
    end
end
Propagate the eikonal solution in the 3D volume;
```
Algorithm 2: Eikonal algorithm with special boundary condition.



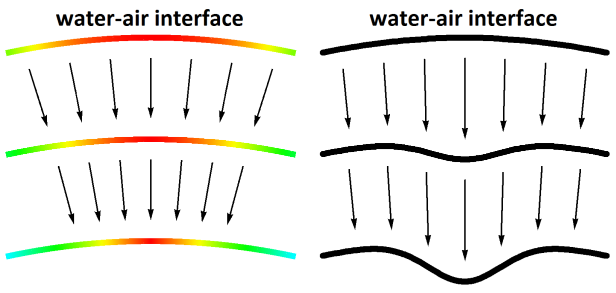
(a) Problem (b) Solution

Figure 3: *Problem caused by the boundary of the domain on the eikonal solution. (a) The eikonal solution cannot be influenced by an interface that lies outside of the domain. The boundary cells will therefore tend to expand as eccentric circles, creating undesired curvature. (b) A solution is to modify the propagation cost at the boundary correspondingly to the angle of incidence of the interface at the boundary.*

eikonal solution a multivalued function. The solution computed by the FMM is the *viscosity* solution, i.e., for every point in space it computes the *shortest* time taken by the wave front to reach this point. This causes a certain loss of information in the solution, which will consequently have an impact on the computed caustics. Figure 5 depicts (in red) what happens to a region of rays coming from the water-air interface when we compute the eikonal solution. The light is concentrated, but since the light rays cannot cross each other, light will accumulate along thin structures in the domain.

To solve this issue, we can apply a post-processing step to the caustic texture. We choose a *threshold height* in the caustic texture where we determine that the rays should have converged. For all the caustic texture values below this height, we conduct a wave simulation in the xy plane to simulate the dissipation of light rays. This is depicted as the blue regions in figure 5.

This post-process requires to choose some parameters, i.e., the threshold height at which we begin to apply the modification, but also the parameters used for the xy wave simulation. Those parameters essentially determine the speed at which light rays dissipate after the threshold height is reached. In figure 5, the parameters determine the angle of the blue cones. We see that even though a given set of parameters might fit perfectly a given region (middle of figure 5), it will likely give incorrect results for other regions (left and right of figure 5). However, the results of section 5.1 show that this rough approximation greatly improves the aspect of the caustics.



(a) Propagation cost distribution (b) Eikonal propagation

Figure 4: *Boundary condition for a 2D boundary of the 3D volume. (a) The propagation cost of interface cells of a 2D boundary are initialized with modified propagation cost. The scale green to red represents high to low propagation costs. This cost is expanded to the other cells of the 2D boundary by performing an eikonal solve using $F = 1$. (b) The interface is then propagated in the 2D boundary using the new modified propagation costs.*

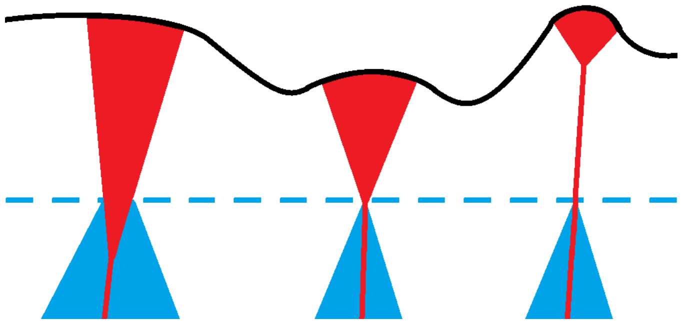


Figure 5: *Effect of the height threshold modification on different light structures. We compare the effect without the height threshold (red) and with the threshold (blue). (left) The chosen height prevents the rays from converging completely. (middle) the height is correctly chosen for this structure. (right) The viscosity solution artefact will be visible for a certain time before the height threshold can dissipate the rays.*

4 Wave Interpretation

In order to eliminate the problems caused by the viscosity solution of the eikonal solution, we study a modified approach. This new method still uses the idea of propagating the water-air interface in the 3D domain, but instead of using the eikonal equation, it uses the wave equation

$$\frac{\partial^2}{\partial t^2}\phi = c^2\nabla^2\phi. \qquad (5)$$

We initialize $\phi = 1$ at cells intersecting the water-air interface and $\phi = 0$ everywhere else. This function serves as the initial condition of the wave simulation. The wave is then propagated iteratively in the domain, and the wave values are accumulated in the 3D cells at each step. The accumulated values are then used as a 3D caustic texture. The process is summarized in algorithm 3.

```
Initialize $\phi = 1$ at interface cells, $\phi = 0$ everywhere else;
for each cell c do
    Initialize causticValue(c) = 0;
end
while not all cells have $|\phi| < \epsilon$ do
    for each cell c do
        CausticValue($c$) += $\phi(c)$;
    end
    Iterate wave simulation;
end
Use CausticValue as the 3D caustic texture;
```

Algorithm 3: The wave algorithm for light front propagation.

The main advantage of using the wave equation instead of the eikonal equation is that it can deal with intersecting light rays. When rays converge, the ϕ values are larger, and they decrease when the light rays diverge, thus yielding larger accumulated values where the light density is higher.

The wave approach also requires special treatment of the boundary, or else the light will reflect on the domain boundary even though this boundary is not physically part of the scene. To avoid such artefacts, we add a band around the domain in which we apply a large damping coefficient to the wave. The damping coefficient is increased progressively as we go deeper in the band, effectively pushing the wave values towards zero without causing ripples in the interior of the domain. This solution is easy to implement, but has the disadvantage that more cells have to be added around the domain, thus increasing the computation size. More sophisticated approaches could be used to impose non-reflecting boundary condition around the domain without using a large additional band of cells [Alpert et al. 2002].

The wave approach is significantly slower that the eikonal approach since it must update the ϕ values in the whole domain for a large enough number of substeps. However, since the wave equation has a finite propagation speed [Evans 1998], the solution only has to be computed in a thin band around the current non-zero values of the solution. This modification significantly improves the computation time by reducing the number of cells that have to be treated during a step to between 5% and 50% of the total cell number.

4.1 Post-Process Modification

The wave equation is more well-behaved than the eikonal equation, i.e., it does not create singularities [Evans 1998]. Even if this is numerically desirable, it is a problem for caustics. Where the eikonal equation can concentrate light intensely in just one grid cell, the wave equation will tend to have a smoother solution and will require a much higher caustic texture resolution in order to have sharply defined caustics.

To palliate this issue, we can perform a post-process step on the caustic texture. Instead of using ϕ as our caustic values, we use $a\phi^b$ where a and b are user defined parameters. Using this transformation can increase the separation between high and low values in the caustic texture, resulting in sharper structures.

5 Results

The following sections present results for the two approaches of sections 3 and 4. In order to have a realistic setting, the water-air interface is taken from a wave simulation on a 2D height field.

Note also that in order to have comparable results, the post-process filter of section 4.1 is applied to all results with different parameters. Those parameters are chosen by hand so that the caustics have the same intensity in each image.

5.1 Eikonal Approach

Figure 7 shows results for the *buried in sand* scene consisting of three objects partially buried in a slope of sand represented by a height field. The figure compares the Fast Marching approach (center and right) to a low resolution photon mapping rendering (left), which serves as a reference for the position of the caustic structures. It also compares the results without (center) and with (right) the post-process step of section 3.3. We use a $256 \times 256 \times 256$ resolution for the caustic texture. Note that our implementation of photon mapping did not allow intersections with arbitrary meshes, so no caustics appears on the three objects.

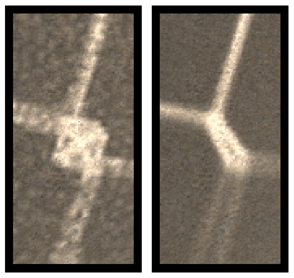
We see that the marching approach creates caustics along the correct curves. The photon rendering shows that the caustics should be diffuse at the top of the slope, then reach a focus height, and be diffuse again at the bottom of the slope. With our threshold height modification, the eikonal approach is able to simulate the correct behaviour in the middle and bottom of the slope. It is not difficult to choose post-process parameters to make those two regions adequately rendered. However, the caustics at the top of the slope are too bright in the marching approach, caused by the rays creating singularities in the eikonal solution too quickly. This could be improved by adding additional post-processing parameters, but there will always be situations that cannot be adequately represented by a simple post-process step.

The marching method presents noise at the top of the slope. This artefact can be reduced by a more accurate initialization of the eikonal solution, but we were not able to make them disappear in all cases. The Fast Marching Method is only first order accurate, which is not precise enough since we compute curvature on its solution, which is a second order operation. Also, since we rely on the singularities of the eikonal equation to produce our caustics, numerical instabilities are to be expected.

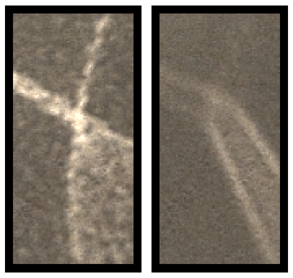
Figure 6 further highlights the differences between the photon and marching with post-process images. Figure 6a shows that the incapability of the eikonal equation to treat intersecting rays turns caustics that should have a certain area into linear structures. Figure 6b shows a region where the threshold height causes at the same time an over- and under-sharpening of caustics structures. This emphasizes the difficulty of choosing this parameter to satisfy all regions of the scene. One possible solution would be to chose different post-process parameters for different regions of the scene, but we have not explored this avenue.

5.2 Wave Approach

Figure 8 shows results for the *pool* scene consisting of an horizontal plane and two vertical planes. The 3D texture has size $128 \times 128 \times 128$. The figure compares a low resolution photon mapping (left) and the eikonal approach (middle) with the wave approach (right). Note that we did not implement photon intersection with the vertical walls.



(a) Since the marching cannot deal with intersecting rays, light areas from photon mapping are turned into linear structures with the marching method.



(b) In some regions, the threshold height smooths the caustic too much, but in others it sharpens it too much.

Figure 6: *Samples from figure 7. For each set, the left image is from the photon mapping image and the right image is from the marching with post-process image.*

We first observe that the caustics given by the wave approach are a lot smoother than in the other two figures. This is due to the fact that the wave simulation is restricted by the finite (and generally low) resolution of the 3D grid. Even though the highlights from the wave approach are located at the brightest regions of the caustics obtained by the other two methods, they do not have the thin appearance we would expect.

The wave simulation is more difficult to control than the two other methods. In one dimension, a wave initialized as a positive bump will propagate as a positive wave localized in space. But in 3 dimensions, the wave does not behave as well and will oscillate between positive and negative values. The propagated wave is thus not localized in time. Modification of the wave equation (e.g. clamping values to $\mathbb{R}^+$) helps to have a more localized behaviour, but introduces errors in the solution. The post-processing values therefore have to be chosen very carefully in order to extract enough information from the wave simulation for the 3D texture to be usable. Note that we do not show the results of the wave approach for the *burried in sand* scene as we could not find parameters that give satisfying results.

Even thought the wave approach is more physically correct than the eikonal approach (because it allows ray intersection), its many issues make it impractical. A more stable way of using the wave interpretation would be to propagate the water-air interface as a 2D surface wave front, much like an iterative beam tracing method.

5.3 Pool Video

The video[1] in the supplementary material shows how the marching method can be used to compute the caustics when the water-air interface is animated. Some frames of this video are shown in figure 1. The interface is initialized as a Gaussian bump and evolved as a wave simulation. The caustic 3D texture is $128 \times 128 \times 128$ and each frame takes about 2.5 seconds to render, which includes the interface wave simulation and the caustic texture computation.

One advantage of the marching approach is that it is purely deterministic (compared to photon mapping where rays are chosen with some level of randomness). This allows to have temporally coherent caustics from frame to frame, and thus doesn't create flickering on the horizontal floor of the pool. Instabilities can be observed on the vertical walls near the water-air interface, but they are due to a poor initialization of the eikonal solution.

[1]Available at `http://youtu.be/sJQs3zDeTpc`

texture resolution	frame rate after texture computation	computing texture with marching	computing texture with wave
64^3	49 fps	0.3 sec	17 sec
128^3	12 fps	2.5 sec	21 sec
256^3	2 fps	26 sec	107 sec

Table 1: *Computational time for the* pool *scene using different resolutions for the caustic texture. We show the frame rate obtained once the texture is computed, the time taken to compute the texture with the marching method, and the time taken to compute the texture with the wave method.*

The animation shows caustics that behave in a reasonable way, especially after the first 30 seconds of the video where the surface reaches a more uniform state that is closer to what we would usually observe in real life.

5.4 Computational Time

The main advantage of using a 3D texture to create caustics is that the same texture can be reused even if we change the scene, as long as the object creating the caustics is not modified. Table 1 shows the time taken by the different methods for different caustic texture resolutions. The second column shows the frame rate obtained once the texture is computed. This is the frame rate at which the user can interact with the scene with a fixed caustic texture. This is only bounded by the efficiency with which our implementation can treat 3D textures. The last two columns show the time taken by both methods to compute the caustic texture.

The wave method is significantly slower than the marching approach. This is because the wave has to compute multiple steps and accumulate values over time in order to obtain the texture values, while the marching method only needs to visit each cell once. Also, the heap structure used by the Fast Marching Method makes the front propagation very efficient.

The times for a $64 \times 64 \times 64$ texture resolution show interactive frame rates, but the obtained resolution is too poor for most applications. The caustics from a $128 \times 128 \times 128$ are better, and the times obtained for this resolution show that nearly interactive frame rates are possible (after the texture has been computed).

Our implementation is in no way optimal, and we believe those times can be significantly reduced. First, the initialization of the wave front from the water-air interface could be improved. In our implementation of both methods, we loop on each vertex of the water-air height field and look at the cells in which it falls to initialize it. This requires a high resolution for the interface, but makes it easy to compute the distance from the interface to a cell center. A more efficient way would be to reduce the water-air resolution and compute the distance by projecting the cell centers on the water-air mesh triangles. Also, the water surface simulation could be parallelized, and the general treatment of 3D textures could be optimized.

6 Future Improvements

Neither the eikonal or the wave approach treat occlusions or shadows, even though this is an important aspect of any realistically illuminated scene. For instance, in the *buried in sand* scene, the three objects should cast shadows on themselves and on the sand. We did not find a simple way to compute shadows from the caustic texture, since the texture is disconnected from the geometry of the scene. Computing shadows directly with, e.g., shadow mapping would not give good results, since this would suppose the light is coming from a single source or a single direction. Caustics are created from light converging from multiple directions, so this assumption is clearly wrong in our case. Since the caustic texture is an indicator of the quantity of light reaching an area, it could possibly be used as a heuristic to modulate the hard shadows obtained by shadow mapping. This is however not trivial and would require additional experimentation.

As pointed out in section 5.1, the Fast Marching Method is probably not the best eikonal solver for our purpose. Since we compute curvature from the eikonal solution, we would like to have a method that is at least second order accurate in space. Second order Fast Marching analogues have been developed [Rickett and Fomel 2000] and could be transparently added to our implementation.

Section 3 also mentions a potential improvement of our method, namely the extension to finite refractive indices. The infinite refractive index approximation used throughout the paper produces acceptable results, but usually causes the rays to converge faster than they usually would in a water and air scene. This extension would be useful to produce more realistic scenes.

7 Conclusion

We presented two different approaches to produce underwater caustics as 3D textures by propagating a wave front from the water-air interface. Even if the wave interpretation is more physically correct and allows intersecting rays, the Fast Marching approach is more efficient and can produce interesting caustics even for low texture resolutions. We were not able to compute the caustic textures at interactive rates in our implementation, but the texture does not have to be recomputed if the scene receiving the light caustics is modified. In that case, we can achieve interactive frame rates.

Our method is too costly and requires too much parameter selection to be efficiently used as a general caustic renderer. The low resolution caustics could still be used, for instance, for a video game application where the water-air interface and its caustic texture can be precomputed.

8 Additional Notes

The entire code for this project was built from scratch using OpenGL and the following libraries : glm, GLFW, FreeImage, AntTweakBar and GLEW. Models for the *buried in sand* scene were taken from Blender. The code is available as a mercurial repository upon request. All ideas for the eikonal and wave approaches, including boundary conditions and post-processing steps, are original ideas of the author.

References

ALPERT, B., GREENGARD, L., AND HAGSTROM, T. 2002. Nonreflecting boundary conditions for the time-dependent wave equation. *Journal of Computational Physics 180*, 1, 270–296.

ARVO, J. 1986. Backward ray tracing. In *Developments in Ray Tracing, Computer Graphics, Proc. of ACM SIGGRAPH 86 Course Notes*, 259–263.

BENAMOU, J.-D. 1996. Big ray tracing: Multivalued travel time field computation using viscosity solutions of the eikonal equation. *Journal of Computational Physics 128*, 2, 463–474.

BRIERE, N., AND POULIN, P. 2001. Adaptive representation of specular light. In *Computer Graphics Forum*, vol. 20, Wiley Online Library, 149–159.

BUSKE, S., AND KÄSTNER, U. 2004. Efficient and accurate computation of seismic traveltimes and amplitudes. *Geophysical Prospecting 52*, 4, 313–322.

DIJKSTRA, E. W. 1959. A note on two problems in connexion with graphs. *Numerische mathematik 1*, 1, 269–271.

EVANS, L. C. 1998. Partial differential equations. graduate studies in mathematics. *American mathematical society 2*.

HECKBERT, P. S., AND HANRAHAN, P. 1984. Beam tracing polygonal objects. In *ACM SIGGRAPH Computer Graphics*, vol. 18, ACM, 119–127.

IHRKE, I., ZIEGLER, G., TEVS, A., THEOBALT, C., MAGNOR, M., AND SEIDEL, H.-P. 2007. Eikonal rendering: efficient light transport in refractive objects. In *ACM Transactions on Graphics (TOG)*, vol. 26, ACM, 59.

IWASAKI, K., DOBASHI, Y., AND NISHITA, T. 2002. An efficient method for rendering underwater optical effects using graphics hardware. In *Computer Graphics Forum*, vol. 21, Wiley Online Library, 701–711.

JENSEN, H. W. 1996. Global illumination using photon maps. In *Rendering Techniques' 96*. Springer, 21–30.

JENSEN, H. W. 2001. *Realistic image synthesis using photon mapping*. AK Peters, Ltd.

LIAO, J., YU, J.-H., AND JIA, L. 2011. Procedural modeling of water caustics and foamy water for cartoon animation. *Journal of Zhejiang University SCIENCE C 12*, 7, 533–541.

LIKTOR, G., AND DACHSBACHER, C. 2011. Real-time volume caustics with adaptive beam tracing. In *Symposium on Interactive 3D Graphics and Games*, ACM, 47–54.

NISHITA, T., AND NAKAMAE, E. 1994. Method of displaying optical effects within water using accumulation buffer. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, ACM, 373–379.

RICKETT, J., AND FOMEL, S., 2000. A second-order fast marching eikonal solver.

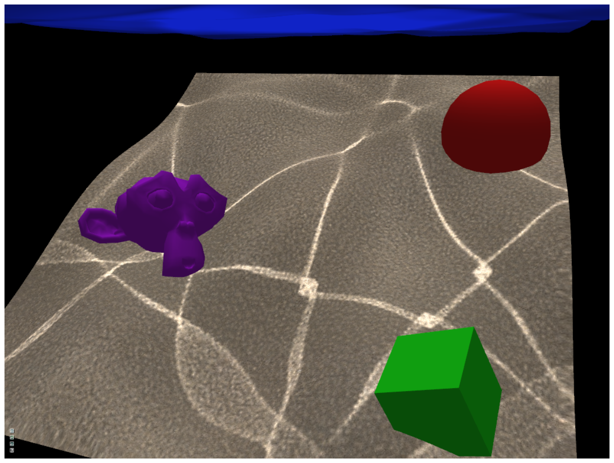
SETHIAN, J. A. 1996. A fast marching level set method for monotonically advancing fronts. *Proceedings of the National Academy of Sciences 93*, 4, 1591–1595.

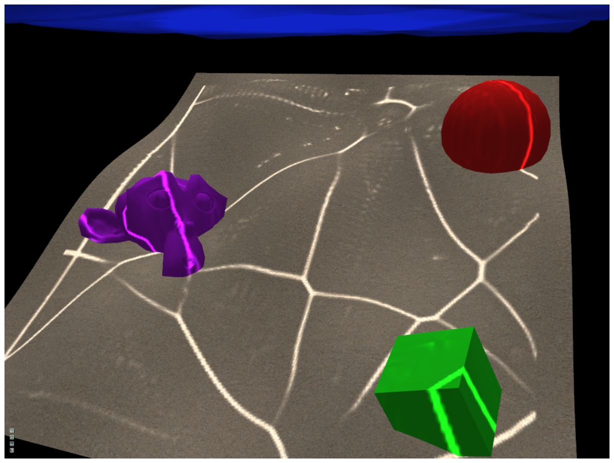
SHAH, M. A., KONTTINEN, J., AND PATTANAIK, S. 2007. Caustics mapping: An image-space technique for real-time caustics. *Visualization and Computer Graphics, IEEE Transactions on 13*, 2, 272–280.

STAM, J. 1997. *Aperiodic texture mapping*. European Research Consortium for Informatics and Mathematics.

WATT, M. 1990. Light-water interaction using backward beam tracing. In *ACM SIGGRAPH Computer Graphics*, vol. 24, ACM, 377–385.

YUKSEL, C., AND KEYSER, J. 2009. Fast real-time caustics from height fields. *The Visual Computer 25*, 5-7, 559–564.


(a) photon mapping


(b) marching approach without post-process

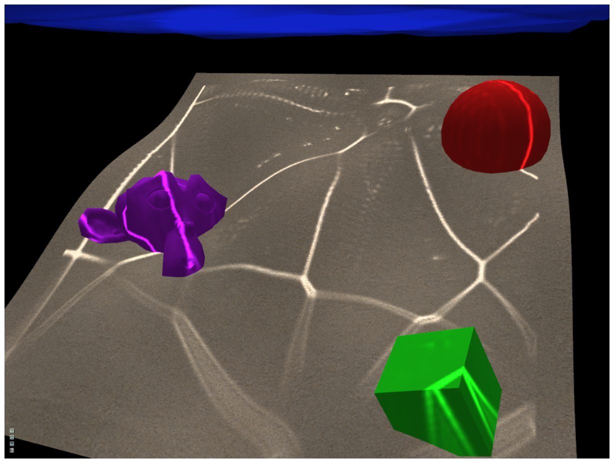
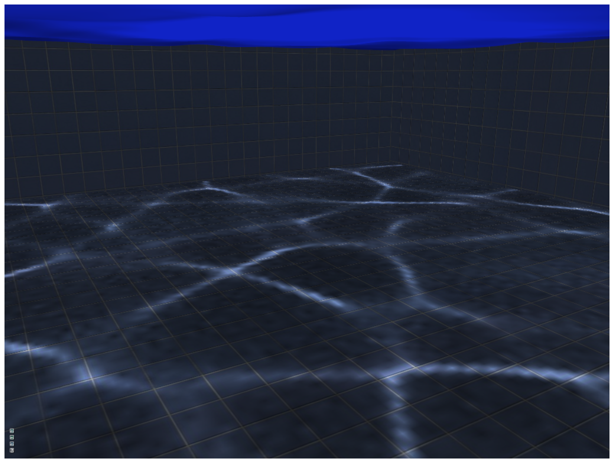
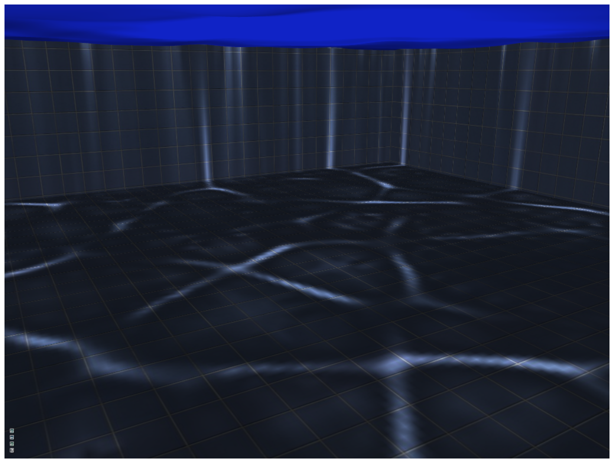

(c) marching approach with post-process

Figure 7: Buried in sand *scene computed with photon mapping and the marching method. The raw marching method creates caustics in the correct regions, and the post processing step is able to correct the width of the caustic structures in some cases.*


(a) photon mapping


(b) marching approach

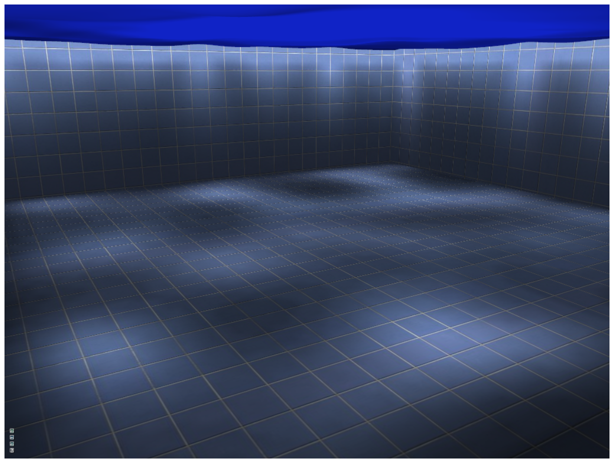

(c) wave approach

Figure 8: Pool *scene computed with three different approaches. The photon mapping and marching approaches give similar results, while the wave method oversmooths the results.*