Virtual Objects on Real Oceans

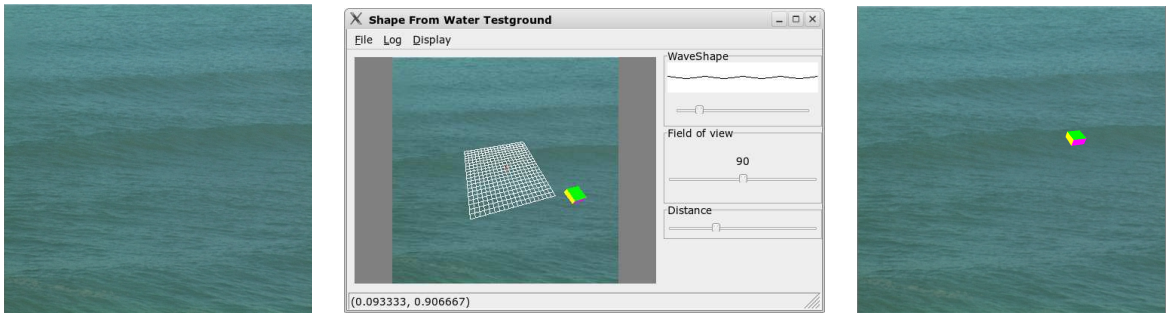
Jean-Christophe Gonzato, Thomas Arcila, Benoît Crespin

Virtual objects on real oceans

J.-C. GONZATO*
Iparla Project (LaBRI - Inria Bordeaux Sud-Ouest)
University of Bordeaux
France

T. ARCILA†
ID Lab (CNRS/INPG/INRIA/UJF)
ENSIMAG - Grenoble
France

B. CRESPIN‡
XLIM
University of Limoges
France



Abstract

Augmented Reality (AR) aims to provide means to integrate virtual objects in a real scene. In that context it is often necessary to recover geometrical information such as objects shapes from the scene in order to add new objects. This paper proposes a semi-automatic method to reconstruct the surface of the ocean from a real ocean scene. A detection algorithm is applied to identify significant waves crestlines. A virtual ocean is then reconstructed using Gerstner model; its parameters are inferred and adjusted by the user to match the crestlines and to provide a smooth reconstruction between adjacent waves. An application is presented to insert a virtual object in the real ocean scene that computes correct occlusions between the ocean surface and the object and uses OpenGL for real-time rendering.

CR Categories: I.3.5 [Computational Geometry and Object Modeling]: Physically based modeling; I.4.8 [Scene Analysis]: Tracking

Keywords: Augmented reality, ocean surface, waves extraction

1 Introduction

AR applications can be seen as an assembly of building blocks, some of which are fundamental components such as tracking, registration or rendering [O.Bimber and Raskar 2005]. In this paper we're interested in the problem of AR applied to the surface of the ocean. The goal is to obtain a 3D model that matches the real surface obtained from a video sequence; this reconstructed model can then be used for various applications: relighting, modification of the surface (to get bigger waves for example), insertion of virtual objects (boats, buoys, etc). In this context, the fundamental components mentioned above have to be addressed, leading us to solve two main problems: tracking the surface and inferring parameter values to reconstruct the model. In order to validate our solution, we present an application that integrates a virtual object in the reconstructed scene, with a correct calculation of the intersection between the object and the surface and real-time rendering.

Several restrictions have to be made due to our acquisition protocol: the video sequence is acquired with only one camera and a unique point of view, and neither markers nor sensors are used. As a prerequisite, we first have to choose an efficient surface model among a large number of models used in realistic image synthesis applications and presented in section 2. The model we preferred, defined by Gerstner, is one of the most simple and efficient but implies that breaking waves or foam appearing in strong wind conditions cannot be taken into account. However, these restrictions help us to simplify the problem while being able to deal with most ocean scenes.

The problem of tracking the surface can then be seen as a computer vision problem, where the goal is to recover geometrical information from a single picture or a sequence of pictures. As a first approach, we could try many "shape from X" algorithms: from texture [Landy and Graham 2004; Loh and Kovesi 2003], from polarization [D. Miyazaki and Ikeuchi 2004; Saito et al. 1999], from focus [Nayar and Nakagawa 1994], from silhouettes [Laurentini 1994], from stereo [D. Scharstein and Zabih 2002], from motion [R. Koch and Gool 2000; Torresani et al. 2004; C. Bregler and Biermann 2000] and finally from shading [J. D. Durou and Sagona 2004]. Unfortunately, due to our acquisition protocol and complex light properties of water material, none of those approaches is usable for our purpose. As an alternative, our detection method presented in section 3 relies on several experiments on image based detection of the ocean surface, and more precisely detection of waves crests. Our algorithm provides satisfying results, although it requires a user intervention which makes it a semi-automatic wave tracking tool.

Section 4 presents our reconstruction method, which aims at obtaining correct parameters of the synthetic wave model from the previous step. The method is applied on each significant wave on the surface; the results are then gathered and blended to reconstruct a single, continuous surface. After the surface is reconstructed, it is rendered as a 3D mesh, and virtual objects can be inserted and rendered in real-time using OpenGL; this step is presented in section

*e-mail: gonzato@labri.fr
†e-mail: thomas.arcila@imag.fr
‡e-mail: benoit.crespin@xlim.fr

5. Finally, our software is detailed in section 6 along with future improvements.

2 Preliminaries on wave physics and hypothesises on video

Two families of algorithms are dedicated to ocean surface simulation in the computer graphics field. The first one is a spectral approach, introduced by Mastin [Mastin et al. 1987]: waves on the surface are represented by their frequency spectrum. This approach computes the wave distribution by a Fast Fourier Transform (FFT) applied on different spectrum types (Pierson-Moskowitz, Hasselmann, etc).

The second family describes the ocean surface by parametric equations. The first work by Fournier and Reeves [Fournier and Reeves 1986] simulates a train of trochoids, relying on both Gerstner and Biesel swell models.

In order to reconstruct a virtual ocean surface from a video, we can notice that the spectral approach is unusable due to its complexity and the "randomness" included in the generated surface. On the contrary, parametric equations are less computationally expensive and can be controlled by very few parameters. We chose a classical model based on parametric equations, developed by the physician Gerstner in 1804, which is described below.

2.1 Gerstner wave model

Let us recall the assumptions of the Gerstner wave model:

- the atmospheric pressure is constant;
- the ocean is composed of a perfect and homogeneous liquid;
- there is no significant wind;
- ocean depth is pseudo-infinite, so there are no frictions with the bottom;
- and finally, there are no ocean currents.

Parameters which describe the wave are its *wavelength* λ, its *amplitude* A, its *curvature* $\gamma = \frac{A}{\lambda}$ and its period T. The *free surface* is the surface of the sea at rest (*i.e.* with no undulations); h is the *depth* of the wave below the surface. These parameters are depicted in Fig. 1. The x-axis is directed horizontally to the beach, the y-axis is horizontal and perpendicular to the x-axis, and the z-axis is vertical.

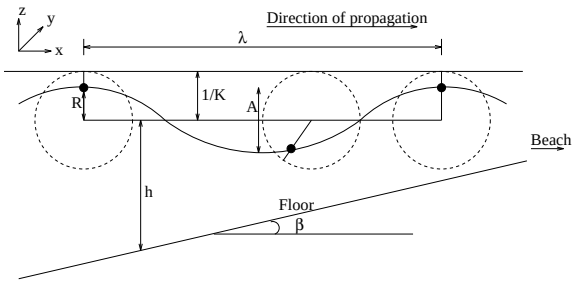


Figure 1: *General definitions.*

The theory proposed by Gerstner describes the motion of a particle of water and is rigorous for an infinite depth. Each particle revolves around a fixed point $M(x_0, z_0)$ which yields a circle of radius R. This circle is included in a disc whose radius is $\frac{1}{K}$, where K is the wave number.

The (x,z) coordinates of each particle are given by:

$$\begin{cases} x = x_0 - R_0 e^{Kz_0} \sin(Kx_0 - \omega t) \\ z = z_0 + R_0 e^{Kz_0} \cos(Kx_0 - \omega t) \end{cases} \quad (1)$$

Parameter ω is called angular speed. The term $R_0 e^{Kz_0}$ is usually simplified to the radius R. In order to obtain the other parameters for a given shape, we can rely on physical laws:

- the wavelength λ is given by $\lambda = \frac{2\pi}{K} = \frac{gT^2}{2\pi}$
- the amplitude $A = 2R$
- the free surface $z_0 = \frac{\pi A^2}{4\lambda}$

The maximal curvature of a wave is obtained when $R = \frac{1}{K}$, so $\gamma_{max} = \frac{A_{max}}{\lambda} = \frac{\frac{2}{K}}{\lambda} = \frac{1}{\pi} = 0,31$.

The product KR characterizes the waves shape as shown on Fig. 2. For our application we need to identify waves crests (or *crestlines*) and troughs – regions between two consecutive crests – from a real ocean video sequence.

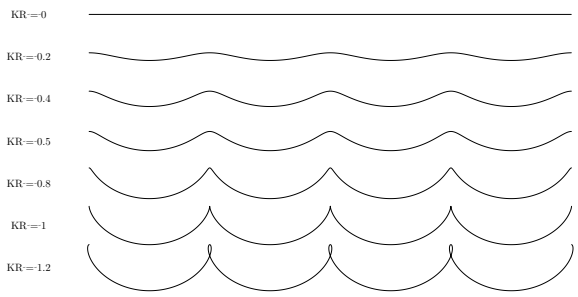


Figure 2: *Shape of waves in function of product KR*

2.2 Hypothesises on video

In order to reconstruct the ocean surface using Gerstner model and as a first approach, we have to make some hypothesis on our video sequence:

- there are no breaking waves,
- there is no foam,
- and finally, no objects (surfers, swimmers, buoys, ...).

These restrictive limitations allow to try to detect wave crests automatically or semi-automatically. Then, we will be able to recover Gerstner model parameters to reconstruct the ocean surface between two consecutive crests.

Another hypothesis on the ocean depth could be discussed, since Gerstner's model is theoretically valid only for an infinite depth, as mentioned above. This would imply that the ocean surface depicted in the video sequence has to be located very far away from the shore. However, in practice the model still gives acceptable results even for limited depth, as long as the wave does not break.

3 Wave detection

The multiple reflections and refractions due to localized, capillary waves generate highly perturbed images; fluctuations in the marine light field are dominated by their variability [Premoze and Ashikhmin 2001]. This problem, along with the fact that the ocean

surface is not a solid, makes it difficult to segment individual images and find similarities between successive images in the video sequence. Usual segmentation methods also fail due to the acquisition protocol itself: the point of view is unique, but the camera isn't calibrated specifically for our application, so that no additional information (exposure time, focal length, etc) can be used. Moreover, the test video sequence is compressed, which introduces artifacts and reduced color interval due to data loss.

However, as described in the next section, to reconstruct a wave our method only needs informations about its crest, which we'll assume is a line in the image; other parameters will be inferred automatically or provided by the user. In this context, we first note that water particles on the crests have greater velocities than particles on the troughs. It is also noticeable that, between two images at time t and $t - dt$, troughs exhibit perturbations, accounting for noisy areas in the image, whereas regions corresponding to crestlines remain relatively coherent.

Since we're interested in detecting crests from significant waves (*i.e.* not capillary waves), the following three-step process is applied:

- each image is first filtered in order to remove noise and visual artifacts due to high-frequency waves;
- it is segmented to detect wave fronts and identify crestlines;
- a user interaction is finally needed to validate the results.

Figure 3: *Reference image*

Image 3 is used in the following as a reference image for our experiments.

3.1 Filtering

Our goal is to keep low frequencies of the image, and remove high frequencies that match capillary waves. Several low-pass filters can be used (median, Gaussian): our experiments showed that the result in any case is far from optimal, since the image becomes too blurry before capillary waves are removed.

An edge preserving filtering strategy such as bilateral filtering [Tomasi and Manduchi 1998] seems more appropriate, since it takes into account color properties by combining a low-pass filter with constraints on color space. It makes use of the Gaussian closeness $gc_{x,y}$ of radius r that allows to estimate similarity between two colors:

$$gc_{x,y} = \exp\left[-\frac{1}{2}\left(\frac{\sqrt{(x-x_0)^2+(y-y_0)^2}}{r}\right)^2\right] \qquad (2)$$

An additional parameter t indicates how filtering should enforce constraints on colors or not. If the distance between two colors is above t, then spatial filtering vanishes; large values of t cause usual Gaussian filtering to be applied. Bilateral filtering of our reference image for $r = 30$ and $t = 15$ is presented on Fig. 4; this result proved to be the best one we could achieve before segmenting the image.

Figure 4: *Bilateral filtering of reference image*

3.2 Segmentation

Our goal is now to extract wavefronts, that is identify regions that represent a single advancing wave. At this stage, one can notice that pixels that belong to wave fronts are globally darker than others. The top lines enclosing those regions will then be identified as crestlines.

Several experimentations were conducted on the filtered image presented on Fig. 4 using simple and straightforward segmentation methods. For instance, applying a *thresholding* operator yields a binary image from which wave fronts could be extracted; Fig. 5 shows an example. Other first-order operators can be used to identify edges, such as Roberts [Roberts 1965], Prewitt [Prewit 1970] or Sobel [Sobel 1970], which all exhibit different properties but are sensitive to noise. Since our input images are noisy but relatively uniform, we were not able to get satisfying results. Finally, another approach to find the positions of wavefronts and crests consists in computing *motion vectors* and their evolution based on the entire video sequence. A motion vector describes a purely translational motion of a selected portion of the image (usually a 16x16 block). The whole set of motion vectors represents the *optical flow*, which can be estimated through predictive coding, namely DPCM [Garcia-Garduno et al. 1994], by taking advantage of highly correlated intensities of neighboring pixels. Thus, the goal is to find similar neighborhoods in successive images. However, results shown on Fig. 6 are not satisfying mainly because optical flow is highly sensitive to image regions that are roughly homogeneous; in that case the optical flow becomes ambiguous. Moreover, assumption about constant intensity in ocean scenes is difficult because of constantly changing transparency, reflections, sparkles, etc. The choice of appropriate parameters is critical to get better results using the operators described above, either to remove isolated pixels and reconnect different parts of a same front, or to estimate motion vectors. We

were not able to get better results although it is theoretically possible.

Figure 5: *Thresholding the reference image at 0.42*

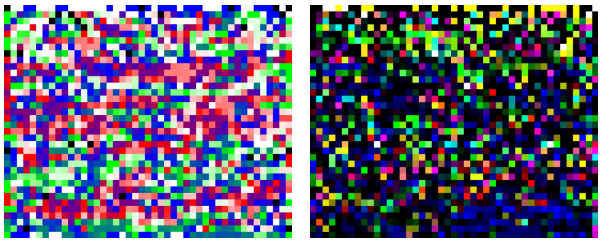


Figure 6: Left: *Block-based estimation of optical flow between two successive images. Each color corresponds to the main direction of the motion vector.* Right: *Evolution of motion vectors on the whole test video sequence. Black blocks represent static motion vectors, i.e. vectors that do not change directions.*

As an alternative to previous methods, we propose to use the intensities histogram of a single image considered as a quadtree. We try to determine if a given cell (ultimately a pixel) belongs to a wavefront or not; if it's not possible, the process is recursively applied to its four subcells. The histogram analysis is summarized below and is applied to the whole image as the starting cell:

1. computation and linearization of the histogram $\mathscr{H}$
2. computation of its mean probability $P = \frac{\sum_{i=1}^{256} H(i)}{256}$
3. detection of significant parts:
   (a) computation of the histogram $\mathscr{H}'$ of the current cell
   (b) convolution of $\mathscr{H}'$ with a low-pass filter to reduce the importance of lower probability regions
   (c) computation of the mean probability P' of $\mathscr{H}'$
   (d) truncation of all values of $\mathscr{H}'$ below the mean value P' to 0
   (e) computation of the minimum and maximum bounds of the remaining values
   (f) comparison of these bounds with the global mean probability P:
      - if both values are lower than P, then the cell belongs to a wave front
      - if both values are greater than P, then the cell does not belong to a wave front
      - otherwise, the cell is subdivided and step 3 is applied to each subcell.

The result, illustrated on Fig. 7, is a label field x which has different values for regions that belong to a wavefront, regions that do *not* belong to a wavefront, and undetermined regions. Although this adhoc method is somewhat similar to a segmentation process based on pixels intensities (dark regions are wave fronts, bright regions are not, and average regions are undetermined), it proved to give much better results than the previous approaches. Despite noisy input data, wave fronts are clearly identified, although the wave at the top was lost as seen on Fig. 7. Moreover, this method is rather simple for single images, which makes the implementation straightforward. However, the method could be improved by considering the whole video sequence and other aspects such as automatically removing possible sky regions in input images. Other methods could also be considered such as Canny edge detector, multiscale analysis or relaxation labeling [Morozov 2008].

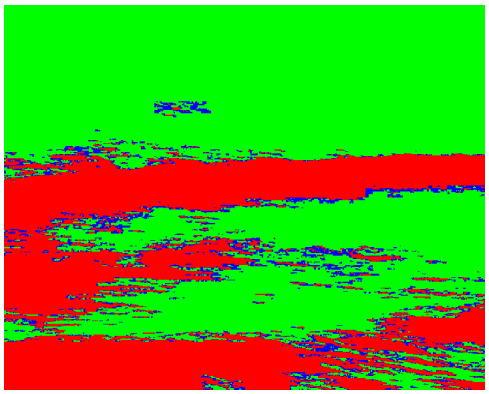


Figure 7: *Segmentation by the quadtree method. Red: regions that belong to a wavefront. Green: regions that do not belong to a wavefront. Blue: undetermined regions.*

3.3 User interaction

At this time, our software lets the user have the possibility to correct crests extraction by adding and/or modifying crest lines. A specific crest can be determined semi-automatically by following and averaging the green/red frontier. The choice of the selected frontier depends on the wave propagation determined by the user. The output, at this step of the process, is a list of crests for each image.

Although it's necessary to perform the extraction process on successive images, it is worth noticing that global parameters (described in the next section) of the video sequence can be reused. This makes our targeted-specific application more useful to segment ocean scenes than general image processing software where this information is not available.

4 Surface reconstruction

4.1 Gerstner model reconstruction

In order to reconstruct the ocean surface for each frame, the user has to provide a shape factor KR which may correspond to the segmented scene (see Fig. 2). Then we need to determine:

- the wavelength λ of each wave train;
- the plane corresponding to the free surface;

- an estimated height of waves H. For this, we will use the shape factor KR. Since $K = 2\pi/\lambda$, R is given by $R = KR/K$, and $H = 2R$.

4.2 Reconstruction

Our method follows four steps:

1. reprojection of the crest lines obtained in the previous section into the 3D space;
2. sorting waves according to their relative position x;
3. reconstruction of each wave by applying Gerstner model;
4. reconstruction of the whole 3D surface by combining independent waves

4.2.1 Reprojection

We now consider the plane S corresponding to the free surface, defined by $ax+by+cz+d=0$, and a projection point p centered on the origin. The projection matrix [M. Woo and Schreiner 2000] is given by:

$$\mathscr{M}_{\mathscr{P}} = \begin{bmatrix} -d & 0 & 0 & 0 \\ 0 & -d & 0 & 0 \\ 0 & 0 & -d & 0 \\ a & b & c & 0 \end{bmatrix}$$

To recover the position of each point in the 3D space, we need to estimate the distance from the camera to the free surface, the inclination of the free surface and finally the field of view of the camera. As a first approximation and if the camera is far from the ocean, we can consider that the real surface is "flat" on the free surface.

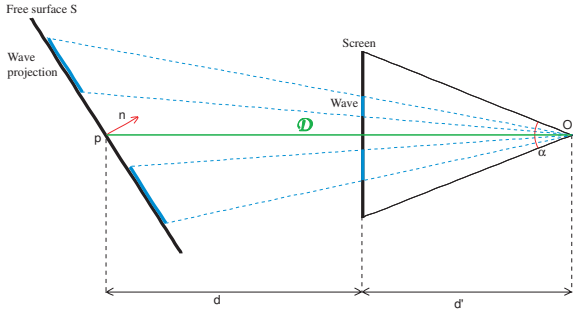


Figure 8: *Reprojection of the viewed waves on the free surface*

Fig. 8 shows the reprojection of the viewed waves on the virtual free surface defined by a 3d point $p(r,s,t)$ and its normal vector $\vec{n}$, depending on point of view $o(x,y,z)$ and field of view α.

Consider a tetrahedron $\mathscr{R} = (o,\vec{i},\vec{j},\vec{k})$, with $\vec{k} = \frac{\vec{op}}{|\vec{op}|}$. The plane S defined by $p(r,s,t)$, with $|\vec{op}| = d+d'$ and $\vec{n}(a,b,c)$ can be described by $S = ax+by+cz-(ar+bs+ct) = ax+by+cz-(\vec{n}\cdot p)$. Thus, the reprojection matrix is defined as:

$$\mathscr{M}_S = \begin{bmatrix} -n\cdot p & 0 & 0 & 0 \\ 0 & -n\cdot p & 0 & 0 \\ 0 & 0 & -n\cdot p & 0 \\ a & b & c & 0 \end{bmatrix}$$

4.2.2 Sorting each wave

After the reprojection step, we have to sort the different waves in order to know the relationship between wave crests and to reconstruct the real 3D surface. We consider the waves on the free surface plane. As waves could propagate in different directions, we compute a global direction $\vec{b}$ which is the average vector of all waves directions only if the divergence is low. Otherwise, each wave is treated separately. In most of cases, we can then deduce a new right-hand frame $(o,\vec{a},\vec{n},\vec{b})$ where $\vec{n}$ is the normal vector to the free surface and $\vec{a} = \vec{n}\wedge\vec{b}$

Each end of waves designed by the user allows to cut the plane into portions along $\vec{b}$. Each deduced space of the plane could contain one or more wave crests and will be treated differently. Fig. 9 shows one decomposition of the free surface by wave crests.

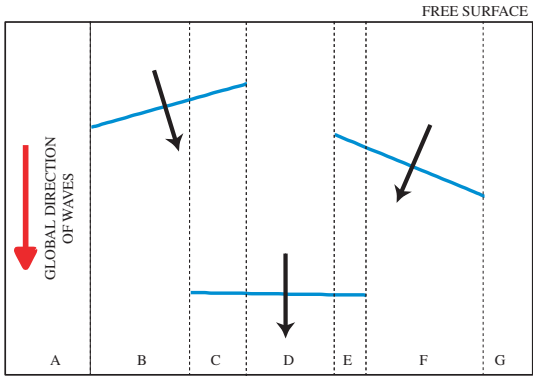


Figure 9: *Three wave crests partitioning the free surface*

4.2.3 Gerstner wave reconstruction

In order to reconstruct the ocean surface with Gerstner model, the user has to design, using our software, the shape of all waves *i.e.* the value of KR (see section 2.1). The idea is to rely on the crests; from the wavelength between two consecutive crests in the same portion of the free surface, we can deduce all parameters of the Gerstner equation, *i.e.* K and R, since $K = \frac{2\pi}{\lambda}$ and KR is specified by the user.

We can reconstruct each trough independently from each others. Thus, we compute the values of K and R for each trough and apply Gerstner model on it.

Union of all independent reconstructed waves.

Unfortunately, with this type of reconstruction, three main artifacts could appear. First, if in a portion of the free surface only one crest is detected (e.g. region B in Fig. 9), the values of K and R are computed by looking at the same crest in neighboring regions. Second, as two consecutive troughs have possibly different values of K and R, the height of the wave crest will be different on the left-hand side and the right-hand side of the crest (see Fig. 10). Third, as we decompose the free surface into separate spaces, some gaps on the surface could appear between adjacent regions.

In order to solve these problems, we multiply each separate reconstructed surface by a mask whose value starts at 1 at the crest and linearly decreases to 0 along the wavelength of the reconstructed portion (see Fig. 11). Each mask covers a part of neighboring regions in order to smoothly blend the final ocean surface. Since the sum of two masks equals one, the reconstructed surface is continuous.

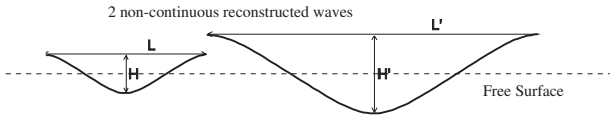


Figure 10: *Errors using our separate surface computation model*

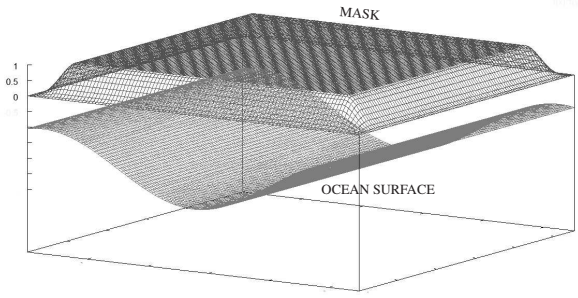


Figure 11: *Mask applied on the surface to solve artifacts*

5 Incrustation of virtual objects

In the previous section, we presented our method to reconstruct a 3D surface of the visible ocean. We use OpenGL to render only the visible part of virtual objects floating on this surface. Finally, we mix the OpenGL rendering with the initial image issued from the video.

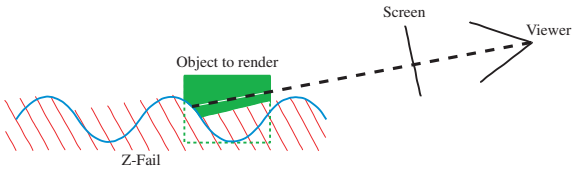


Figure 12: *Image of the Z-Buffer with only the part of the visible cube over the surface.*

We have to treat two cases: the first one when the ocean surface is considered as an opaque material and the second one in the case of a semi-transparent surface.

Opaque surface The method consists in rendering only the visible part of the virtual objects over the surface, then mix it with the original image. The algorithm can be decomposed into three steps. First, we render the original image issued from the video. Second, we render the 3D model (ocean surface) by disabling display and enabling only Z-buffering. At this step, we know which parts of the original image are occluded by waves. Finally, we render only the virtual objects: only their visible parts will be displayed, thanks to the previous Z-buffer test.

Transparent surface Most water surfaces could be rendered without transparency since the viewer is usually too far. However, our method is also able to treat specifically the transparency over immersed parts of virtual objects.

The algorithm follows the same steps as for opaque surfaces *i.e.* render the original input image, render the virtual surface only into the Z-buffer, and finally display virtual objects on the original image using the Z-buffer test.

The difference for semi-transparent ocean surface consists in activating the stencil buffer during the virtual objects rendering step. When the object is immersed, the Z-fail is verified and we set the corresponding pixels to 1 in the stencil buffer. Finally, we re-render the original image with transparency enabled only where the stencil buffer equals 1.

6 Results and future works

6.1 Results

Starting from the reference image, our software is able to analyze the scene and retrieve semi-automatically all the wave crests visible in the image. The user has to validate the shape of each wave, and define the distance between the camera and the free surface of the ocean, the virtual free surface and finally the field of view of the camera. A snapshot of the GUI is presented on the first page of this article.

The position of the free surface is not so easy to define. In fact, the user has to take into account the distance of the free surface from the camera and its orientation. No information can be retrieved from the video, however the user can easily find a satisfying empirical solution.

One of the difficulties is to find a reconstructed model as coherent as the reality. In the same manner, it is difficult to automatically match the scale of virtual objects with the real surface. To solve these two problems, we only propose a visual validation.

Figs. 13.a-b show different position of a virtual object on a real image. Fig 13.c shows teapots on different position on the ocean surface. We need to add reflection of the objects on the surface in order to have a real incrustation. Fig. 13.d shows the position of the virtual free surface of the waves represented as a grid. The computation time is quasi instantaneous (analysis + rendering) on a classic personal computer, although only simple 3D objects are incrusted in our tests.

At present time, we can render in real time many virtual objects on a still image of the ocean scene. We can also drag, in real time, one object along the surface in order to see it passing back and forth on each side of the wave.

6.2 Future works

The presented method is a first approach of a more general problem we would like to solve : first, how to incrust any object on various types of ocean scenes, scenes with foam and spray, with a wide range of objects (surfers, swimmers, buoys, ...). Second, how the incrusted objects could interact and modify the original images locally.

The next steps of our researches will deal with :

- the enhancement of the automaticity of our algorithm by changing the type of the video camera (presently mini-DV), and by accessing EXIF parameters (Field of View, angles, distance from the ocean surface, ...);
- taking the whole video into account in order to catch temporal coherence and remove parasitic objects (foam, swimmer, ...)
- retrieving the luminance of the real surface point to point depending on the virtual surface in order to modify it locally and reconstruct a modified real surface, using for instance, stochastic motion textures [Chuang et al. 2005];

- considering other methods such as Canny edge detector, multiscale analysis or relaxation labeling to detect the wave crests automatically.

7 Conclusion

In this article, we have shown how to build a 3D model of the surface of the ocean, using a simple video sequence and user-provided information. This two-step process relies on two different algorithms: a wave detection step that identifies different waves in the scene, and a reconstruction step to recover the parameters of the model for each wave, which are then blended to obtain a single, continuous surface. The model is then integrated along with a virtual object directly in the video sequence.

The proposed GUI is very simple to use and allows to get acceptable results very fast, since the wave model we chose is defined by very few parameters. Another wave model could also be used to take into account breaking waves and other complex, non-linear phenomena. The same work done with picture specialized software would take much more time to achieve the same quality.

Concerning the detection step, our first idea was to automatically extract information about waves from the video sequence but we faced several difficulties due to a poor-quality acquisition protocol, noisy video and compression artifacts. Even though user intervention is still necessary, image-based detection gave good results. We believe that a fully automatic method is possible, for example by taking advantage of the temporal coherence to initialize parameters such as the position of the free surface and the direction of the waves only once, and use inter-frame coherency to detect waves.

An application to insert a virtual object in the scene was implemented but, since they're not physically-based, the results still look too "synthetic". They could be greatly enhanced by fully simulating fluid / rigid body interactions, although this would require a significant amount of computations, for example if we wish to render the wake of a virtual boat cruising on the real ocean.

The rendering process would also prove itself more useful if it was able to provide better realism by integrating shadows and complex light-water interactions: reflection, refraction, caustics, etc.

References

C. BREGLER, A. H., AND BIERMANN, H. 2000. Recovering non-rigid 3d shape from image streams. *IEEE Conf. Computer Vision and Pattern Recognition 2*, 690–696.

CHUANG, Y.-Y., GOLDMAN, D. B., ZHENG, K. C., CURLESS, B., SALESIN, D. H., AND SZELISKI, R. 2005. Animating pictures with stochastic motion textures. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Papers*, ACM, New York, NY, USA, 853–860.

D. MIYAZAKI, M. K., AND IKEUCHI, K. 2004. Transparent surface modeling from a pair of polarization images. *IEEE Transactions On Pattern Analysis And Machine Intelligence 26*, 1 (January), 73–82.

D. SCHARSTEIN, R. S., AND ZABIH, R. 2002. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International Journal of Computer Vision 47*, 7–42.

FOURNIER, A., AND REEVES, W. T. 1986. A simple model of ocean waves. In *Computer Graphics (SIGGRAPH '86 Proceedings)*, D. C. Evans and R. J. Athay, Eds., vol. 20, 75–84.

GARCIA-GARDUNO, V., LABIT, C., AND BONNAUD, L. 1994. Temporal linking of motion-based segmentation for object-oriented image sequences coding. In *Proc. of the VII European Signal Proc. Conf. (EUSIPCO'94)*, 147–150.

J. D. DUROU, M. F., AND SAGONA, M. 2004. A survey of numerical methods for shape from shading. Tech. rep., IRIT, Université Paul Sabatier, January.

LANDY, M. S., AND GRAHAM, N. 2004. Visual perception of texture. *The Visual Neurosciences*, 1106–1118.

LAURENTINI, A. 1994. The visual hull concept for silhouette-based image understanding. *IEEE Trans. Pattern Anal. Mach. Intell. 16*, 2, 150–162.

LOH, A., AND KOVESI, P. 2003. Estimation of surface normal of a curved surface using texture. *Proceedings Of The Digital Image Computing*, 155–164.

M. WOO, J. NEIDER, T. D., AND SCHREINER, D. 2000. *OpenGL programming guide*. Campus Press.

MASTIN, G. A., WATTERBERG, P. A., AND MAREDA, J. F. 1987. Fourier synthesis of ocean scenes. *IEEE Computer Graphics and Applications 7*, 3 (Mar.), 16–23.

MOROZOV, M. A. 2008. Tracking of sea and air flows from sequential satellite images by the relaxation-contour method. *Pattern Recognition and Image Analysis 18*, 1, 107–111.

NAYAR, S. K., AND NAKAGAWA, Y. 1994. Shape from focus. *IEEE Trans. Pattern Anal. Mach. Intell. 16*, 8, 824–831.

O.BIMBER, AND RASKAR, R. 2005. *Spatial Augmented Reality*. AK Peters.

PREMOZE, S., AND ASHIKHMIN, M. 2001. Rendering natural waters. *Computer Graphics Forum 20*, 4, 189–200.

PREWIT, J. 1970. Object enhancement and extraction. In *Picture Processing and Psychopictorics*, 75–149.

R. KOCH, M. P., AND GOOL, L. J. V. 2000. Realistic surface reconstruction of 3d scenes from uncalibrated image sequences. *Journal Of Visualization And Computer Animation 11*, 3, 115–127.

ROBERTS, L. G. 1965. Machine perception of 3d solids. *Optical and Electro-optical Information Processing*, 159–197.

SAITO, M., SATO, Y., IKEUCHI, K., AND KASHIWAGI, H. 1999. Measurement of surface orientations of transparent objects using polarisation in highlight. *IEEE Conference On Computer Vision And pattern Recognition 1* (June), 381–386.

SOBEL, I. E. 1970. *Camera models and machine perception*. PhD thesis, Standford University.

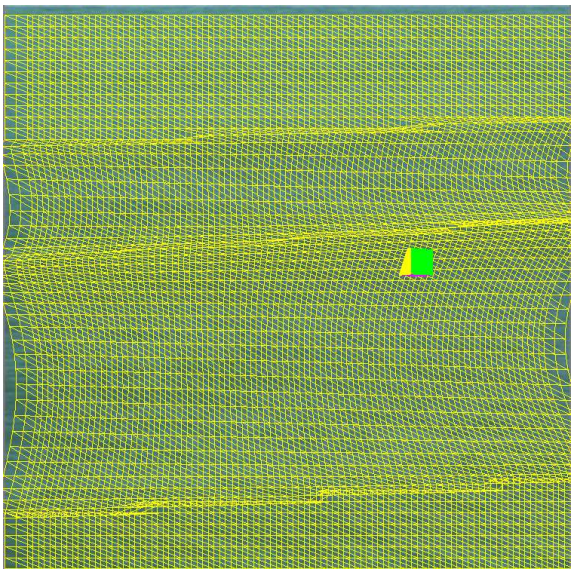
TOMASI, C., AND MANDUCHI, R. 1998. Bilateral filtering for gray and color images. *IEEE International Conference on Computer Vision*, 839–846.

TORRESANI, L., HERTZMANN, A., AND BREGLER, C. 2004. Learning non-rigid 3d shape from 2d motion. *Advances in Neural Information Processing Systems*.

a. Object incrusted in front of wave

b. Object behind the wave

c. Objects and Ocean



d. Virtual reconstructed surface

Figure 13: *Results...*