Ocean Surface Rendering Using Temporal Texture Synthesis Technique

Chao-Hung Lai
Department of Computer Science and Engineering,
National Chung Hsing University
phd9415@cs.nchu.edu.tw

Jiunn-Lin Wu
Institute of Networking and Multimedia,
Department of Computer Science and Engineering
National Chung Hsing University
jlwu@cs.nchu.edu.tw

***Abstract*—A new algorithm for rendering ocean surface using texture sampling and height field superposition techniques is presented. The proposed method requires two static images as input, one for reproducing the detailed ocean waves and the other for reconstructing the main structure waveform. The input images are first used to synthesize toroidal textures which are then used to reconstruct the height fields. The ocean surface is tessellated straightforwardly with the superposition of the wave height fields with different spatial resolutions. We introduce a simple relighting method to reproduce the wave shading and undulating effects. Experimental results show that the proposed algorithm is convenient and efficient to produce realistic ocean surface.**

***Index Terms*—Temporal texture synthesis, height field, shade-from-shading.**

I. Introduction

Rendering realistic ocean surface is an interesting and important problem in computer graphics. Many researchers have been devoted to producing ocean scenes using mathematically or physically based simulation [14, 15, 17, 18]. These techniques are able to produce photorealistic or non-photorealistic scenes. However, they have to develop particular formulas for specific effects and require expensive computation. Recently, GPU on a graphics card is used to achieve a real-time performance in spectral models [2, 5, 6, 7, 13]. However, most of desktop and notebook computers have integrated GPUs, which are cheaper but usually far less capable than dedicated graphics cards [20]. Without powerful GPU systems, texture mapping is a cheap and good solution to render realistic scenes.

A texture-based method has two primary advantages: First, a texture is easy to obtain. It is convenient to produce ocean surface animations by just giving the photo textures. The other vantage is that textures are portion of natural scenes, without artificial complex computation, which have their intrinsic physical properties. This is the essence of texture sampling technique that what you see is what you get.

Although it is convenient to obtain textures, however, photo textures are limited to their inherent properties, such as color, lighting, shading, and, especially, resolution. To synthesize an ocean surface and apply it in a wide spreading ocean landscape requires solving a number of problems. The first challenge is in synthesizing a large enough surface to cover the rendering resolution according to the current view frustum. The second difficulty is in creating the waving structure without repetitively tiling. In addition to producing the small water waves in quasi-periodic ways, the low-frequency waveform is also needed to exhibit the main structure. The third challenge is in animating an oceanographic motion.

In this paper, we present a method to reproduce ocean surface using temporal texture. The user determines a water image to be used as the appearance of a highly detailed ocean surface. This input image is given to synthesize a higher resolution texture with toroidal property which is used to be the small water waves. The user inputs another image to be used as the low-frequency waveform. This image is given to synthesize a lower resolution texture which is used to be the main wave structure. In order to construct the surface waves, we employ a shape-from-shading technique [4] to recover the height fields of the two wave textures. The whole ocean surface is then tessellated by the composite of the two height fields with different scaling resolutions. The elevation of the surface is computed on the fly and the computational cost is proportional to the visible region of the view frustum. After the composed height field of the surface is determined, we use a simple relighting method to adapt the luminance to render the waving appearance. This method is based on the experimental evidence that observers treat lower luminance values as more concave surface locations [8]. Finally, the main structure animation is done by shifting the low resolution waveform. The animation of small water waves is achieved by the method proposed by [12]. In practice, the waves motion cost can be almost ignore, because the motion is achieved by only adding a translating vector.

The proposed method is not to achieve a physically accurate reproduction of the ocean scenes. The main contribution of this paper is as follows. The ocean surface is realistically synthesized using real world textures which have their intrinsic color and physical properties. The computational cost is cheap for each frame, because only the pixel interpolation and relighting computation are needed, and it is proportional to the number of water surface pixels in the view frustum.

II. Related Work

Most previous researches for rendering ocean surface are mathematical or physical simulation. Their procedures use a sum of sinusoidal amplitudes and phases to model the wave geometry [15, 17]. The following researches are absorbed in wave formulas development such as trochoids [3, 6, 18] or Fourier domain waveforms [2, 7, 13]. These techniques have to determine particular formulas for specific phenomena and require a lot of superposed sinusoidal or spectral functions which are computational expensive.

Recently, many researches have been engaged in rendering motion scenes using texture-based methods. They can be generally categorized into two classes. The first class includes the video-based methods which require a video clip as input sample to synthesize a new video sequence [1, 10, 16, 19]. Wei and Levoy [19] modeled the input video as 3D exemplars. The pixel whose 3D neighborhood is matched by a similarity constraint is copied into the output volume. This copy process proceeds pixel by pixel and slice by slice to synthesize a new video. Schodl et al. [16] proposed a conception of video texture. They analyze the input video by a temporal similarity condition to extract the video texture elements. These elements are recombined to synthesize temporally infinite output video with preserving the similarity across frames and the dynamics of motion. Kwatra et al. [10] proposed a graph cut algorithm. They concatenate volumes of input video with minimum error seams to create infinitely long sequences. Bhat et al. [1] presented a flow-based video synthesis method. They capture the motions of the input video along user-specified flow lines and interactively synthesize seamless video of arbitrary length by enforcing temporal continuity. Using these video-based methods, the user is required to provide the source video, which is less convenient. Besides, the output video is limited to its intrinsic coloring and lighting, which obstructs the further applications for texture mapping.

The second class includes the image-based methods which require only a 2D image as input to synthesize 3D temporal textures. Kwatra et al. [11] extended their texture optimization algorithm to synthesize the texture sequence that moves according to a given flow field. However, this technique is unable to create infinitely long sequence and it is time-consuming to create each frame. Kopf et al. [9] proposed an optimization technique to synthesize solid textures from 2D exemplars. They use a solid optimization method to enforce the local similarities between the synthesized voxel and the exemplar, and introduce a re-weighting scheme by histogram matching to cause the global statistics of the synthesized texture to match the exemplar. Their solid texture results work well in creating solid objects. However, the solid texture is a video loop with finite duration and it is incompetent for rendering the ocean surface with similar, quasi-repetitive properties. Our previous work [12] introduced a temporal texture synthesis algorithm based on transition linking and morphing interpolation techniques. This method requires only a static image texture as input and employs the probabilities of similarity attached with the transition links to generate an inexhaustible sequence with quasi-periodic quality. The results are exactly suitable for rendering the waving ocean surfaces.

III. The Proposed Algorithm

The proposed system can be divided into two stages, including preprocessing and rendering. In the preprocessing stage, the user inputs two static images to build the ocean surface. In the rendering stage, a simple relighting method for enhancing the waving effect and reducing the repetitiveness is used to reproduce the main waveform.

A. Surface texture synthesis

In general, the ocean surface can be assembled from a main structure waveform and small water waves [13, 18]. Our model requires only two water images as input to synthesize an infinitely wide spreading ocean surface. One image is used to synthesize a toroidal texture for tessellating the highly detailed surface, while another is used to build a lower resolution texture for representing the low frequency waveform. The shape of the ocean surface is synthesized using the two height fields built from the two textures respectively.

Synthesizing a high resolution texture for the whole visible surface area requires costly computation. It is more efficient to synthesize a lower resolution texture with seamless tileable property to tessellate the visible region according to the view frustum. But the tessellation will produce noticeable repetitive patterns. Therefore, we synthesize another tileable texture as an overlapping wave height field of the surface. We use the chessboard filling texture synthesis procedure proposed in our previous work [12] to synthesize the tileable texture for the main structure waveform. For animation effect, we use the temporal texture synthesis method proposed in [12] to synthesize the dynamic texture for small water waves. This previous method requires only a static texture image as input to synthesize plausible sequences which can be infinitely extended in both space and time. This output can exactly be used to present the motion of water waves. Figure 1 shows a synthesized example for small waves and main structure.

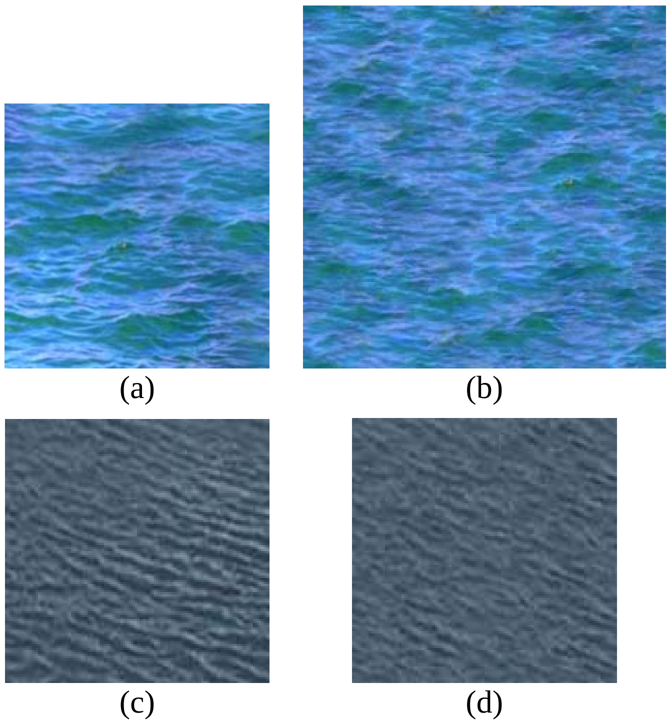


Figure 1. Tileable texture synthesis. (a) Input texture for small waves. (b) A synthesized result of (a). (c) Input texture for main structure. (d) A synthesized result of (c).

For illustration, we briefly describe the proposed temporal texture synthesis algorithm presented in our previous work [12]. Please refer to the original work for more details. The basic concept is that given a 2D texture image as input, we annex the temporal component to the synthesizing process to produce 3D temporal textures. At first, the proposed basis sequence generation procedure is used to synthesize a plausible sequence of images. Figure 2 illustrates the process of synthesizing each frame in the sequence. Both the spatial and temporal constraints are considered to search for matched patches in the synthesizing process. That is, the matched patches must yield to the similarity of spatial neighborhoods (O-shaped boundary zone), and meanwhile, they must yield to the similarity of temporal neighborhoods which are the patches at the same position in the previous frame. Besides, all the frames in the sequence are seamlessly tileable because the patch boundary zones are treated toroidally.

At the next stage, we measure the similarity of in-between frames and build explicit frame-to-frame transition links for reorganizing the order of the frames with smooth visual realism. And then, the proposed automatic morphing technique is used to interpolate smooth metamorphosis between each pair of frames connected by the transition links. Figure 3 illustrates the interpolating process, in which, for example, frame F_i and F_j are warped to each other by computing the affine mesh mapping and both are blended together to produce the intermediate morphing versions. The reachable complexion is guaranteed that each frame may have the path to another frame by walking through certain of transition links which are interpolated using natural metamorphosis.

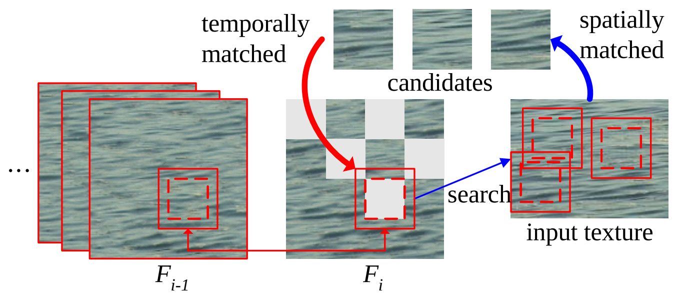


Figure 2. Spatial and temporal constraints. The candidates in the set of spatially matched patches are then extracted by the similarity with the patch at the same position in the previous frame.

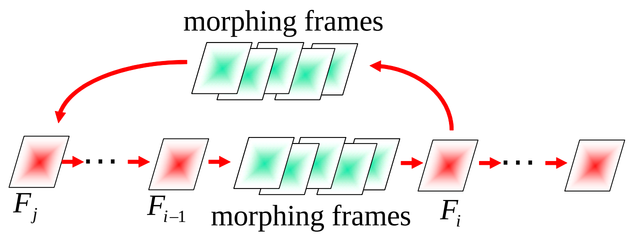


Figure 3. Interpolating smooth metamorphosis into each linked transition.

B. Height field building and surface relighting

For rendering the undulation effect, it is desired to reproduce the height field of the surface. The elevation of the surface is composed of the height fields of the main structure waveform and the small water waves. We borrow the shape-from-shading method presented in [4] to reconstruct the height fields of every image including the main structure texture and all the nodes in the linked sequence for small waves. The height field reconstruction formula is as follows.

$$height(x,y) = hf \cdot \frac{I(x,y) - I_{\min}}{I_{\max} - I_{\min}}, \qquad (1)$$

where hf denotes the height factor, $I_{\max}$ and $I_{\min}$ denote the maximum and minimum intensity for each image, and $I(x,y)$ is the intensity at position (x,y).

For elaboration, let M_h denote the height field derived from the main structure texture. Let P be the pointer pointing to a node in the small waves linked sequence, P_c be the image attached by P, and P_h be the corresponding height field. Initially, the color $S_c(x,y)$ and height $S_h(x,y)$ at a position (x,y) on the ocean surface is constructed as follows.

$$S_c(x,y) = P_c(x,y), \qquad (2)$$

$$S_h(x,y) = P_h(x,y) + M_h\left(\frac{x}{scale}, \frac{y}{scale}\right), \qquad (3)$$

where $scale$ is a scaling factor.

To reproduce the main structure waveform, we propose a simple relighting method. Theoretically, Eq. (4) and Eq. (5) show the relation between $I(x,y)$ and $height(x,y)$ derived from Eq. (1).

$$I(x,y) = I_{\min} + \frac{height(x,y)\cdot(I_{\max} - I_{\min})}{hf}, \quad (4)$$

$$\Delta I(x,y) = \frac{\Delta height(x,y)\cdot(I_{\max} - I_{\min})}{hf}, \quad (5)$$

where $\Delta height(x,y)$ and $\Delta I(x,y)$ denote the alteration of $height(x,y)$ and $I(x,y)$ respectively. Trivially, the intensity can be modified according to the variation of height value and the resulting $I'(x,y)$ is as follows.

$$I'(x,y) = I(x,y) + \Delta I(x,y), \quad (6)$$

However, it must be quaranteed that the resulting $I'(x,y)$ is set in the valid range $[0,255]$. So, we modify Eq. (5) as follows.

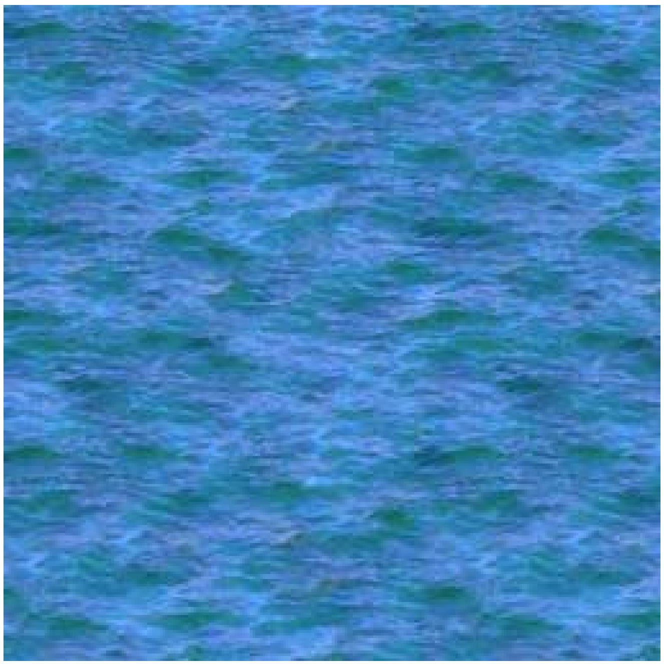
$$\Delta I(x,y) = \Delta height(x,y)\cdot dhfac, \quad (7)$$

$$\Delta height(x,y) = M_h(\frac{x}{scale}, \frac{y}{scale}) - \overline{M_h}, \quad (8)$$

$$dhfac = \begin{cases} \left(\frac{255 - I(x,y)}{\Delta height_{\max}}\right)^{\frac{3}{4}}, & if \quad \Delta height(x,y) \geq 0 \\ \left(\frac{I(x,y)}{\Delta height_{\max}}\right)^{\frac{3}{4}}, & if \quad \Delta height(x,y) < 0 \end{cases}, \quad (9)$$

where $\overline{M_h}$ is the average height of the main structure field, and $\Delta height_{\max}$ is the maximum delta height. We experimentally set the exponent as $\frac{3}{4}$ to make the intensity vary appropriately and the result is satisfactory.

For each frame, the computational cost of the surface comes from the interpolating calculation of $M_h(\frac{x}{scale}, \frac{y}{scale})$ and relighting. The ocean surface is tiled using the composition of the height fields of main structure and small waves at different scales. After relighting, the repetitiveness is unnoticed. Figure 4 shows the ocean surfaces before and after relighting.


(a)

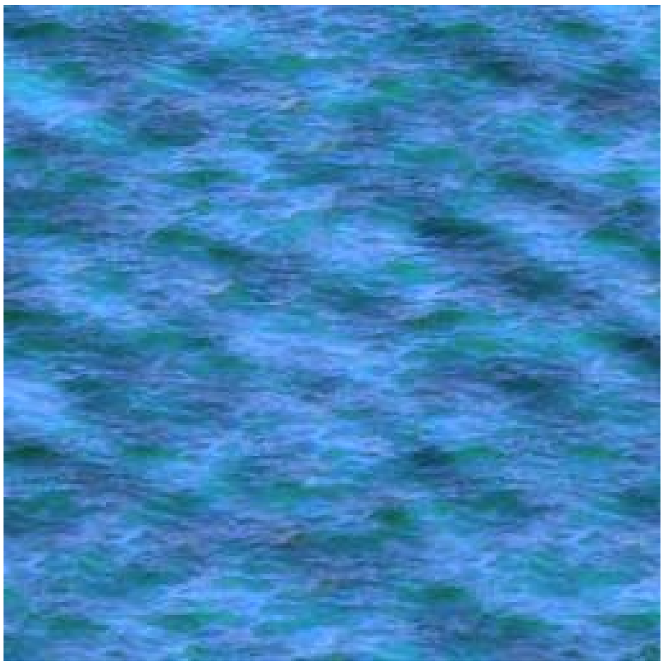

(b)

Figure 4. An ocean surface example before and after relighting. (a) Before relighting. (b) After relighting using the main structure shown in Figure 1(d).

C. Rendering

For realistic rendering the undulating effect, a ray-marching process is preferable [3, 18]. However, ray-marching is the bottleneck process of rendering realistic ocean surface. Reducing the computation of ray-marching will promote the whole performance. In this work, the texture-based surface after relighting has its inherent color and shading. From this point of view, we can ignore the height variation and turn the ray tracing process to the trivial perspective projection, as illustrated in Figure 5. This simple scheme can perform fast rendering.

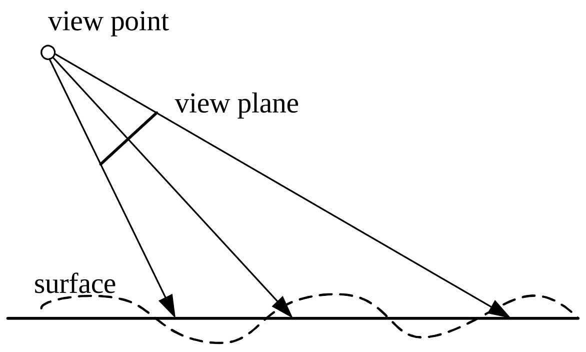


Figure 5. Simple perspective projection. The surface is considered as a plane.

D. Animation

The animations of main structure waveform and small water waves are considered separately. The main structure animation is simply done by adding a shifting vector. We animating the small waves by using the temporal texture rendering method presented in [12]. For each frame, the height field and color of the surface is computed as follows.

$$S_h(x,y) = M_h\left(\frac{x+t\cdot m_x}{scale},\frac{y+t\cdot m_y}{scale}\right)+ P_h(x+t\cdot s_x, y+t\cdot s_y), \quad (10)$$

$$S_c(x,y) = P_c(x+t\cdot s_x, y+t\cdot s_y)+ relighting(x,y), \quad (11)$$

where t is the time parameter, (m_x, m_y) and (s_x, s_y) are the shifting vector added into main structure and small waves respectively. The pointer P will keep pointing to the next appropriate node according to the current node and the associated similarity probability. Consequently, except for the rendering of each frame, the animation process almost does not cost any more computation.

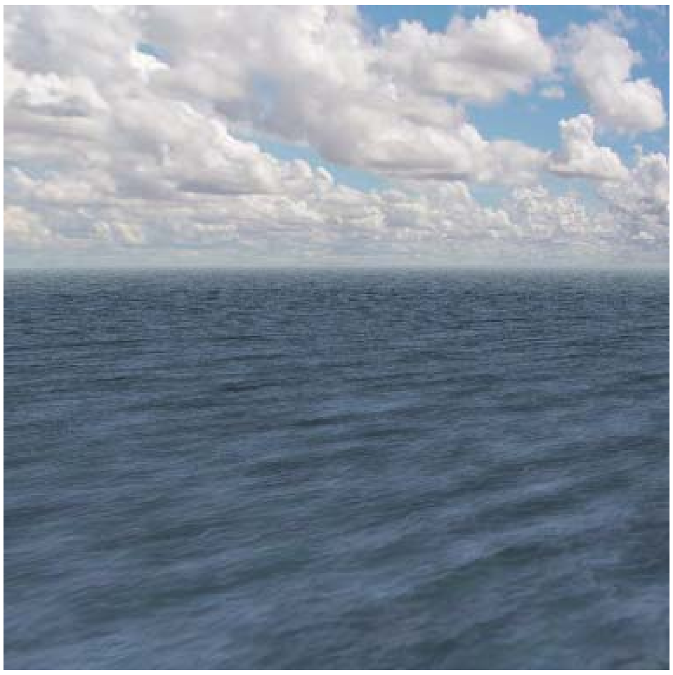


Figure 6. An result for ocean scene.

IV. Experimental Results

The proposed method requires two static images as input for reproducing the ocean surface. In the preprocessing stage, we synthesize the toroidal textures and reconstruct the height fields for main structure waveform and small water waves. In the rendering stage, we use the relighting method to enhance the waving effect and reduce the repetitiveness and use the trivial perspective projection method to perform rendering. We implement our algorithm using Microsoft Visual C++ 6.0 on a computer with Intel Core2 1.8 GHz CPU and 1 GB RAM. Figure 6 shows a result, in which the waves are moving toward south west direction. The frame resolution is 384×384 and the rendering time for each frame is about 0.187

seconds. Our interface allows the user to change the view frustum, including the location of the view point, and the location, direction, and resolution of the view plane. Figure 7 shows another result with different ocean surface and frustum direction.

Figure 8 depicts another example. We take a portion from the ocean part of a real world image shown as Figure 8 (a) and use this portion as the input texture shown as Figure 8 (b) for building the small waves. The main waveform is built from Figure 8 (c). Figure 8 (d) shows the plausible rendering result. The frame resolution is 384×384 and the rendering time is about 0.281 for each frame. This rendering time is longer than the result shown in Figure 6 because there are more ocean surface pixels in Figure 8 (d). The computational cost is cheap and it is proportional to the number of surface pixels in the view frustum.

In practice, we can easily compute the normal vector for each water pixel in the view frustum. After the normal vectors are determined, the refraction and reflection effects can be handily simulated. Figure 9 shows a refraction result with a rock texture as the bed. As shown above, the proposed algorithm is successful to produce the sequences which resemble realistic ocean scenes.

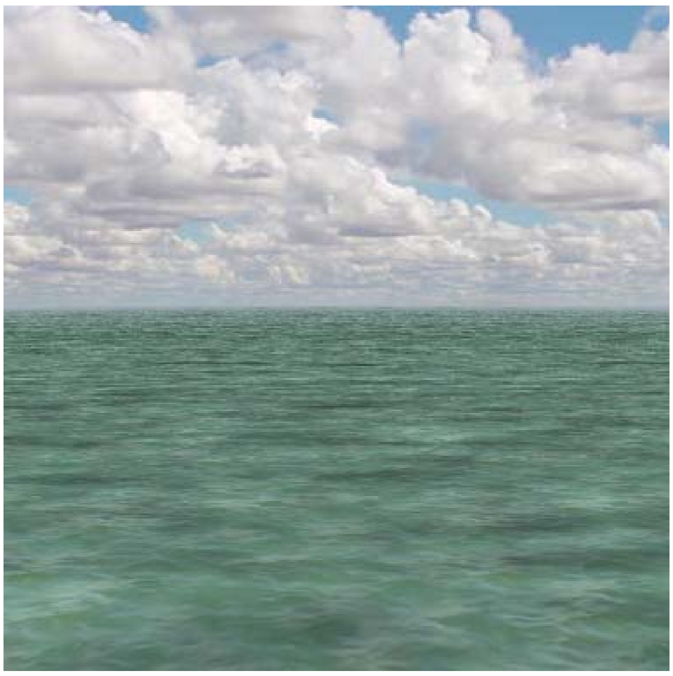
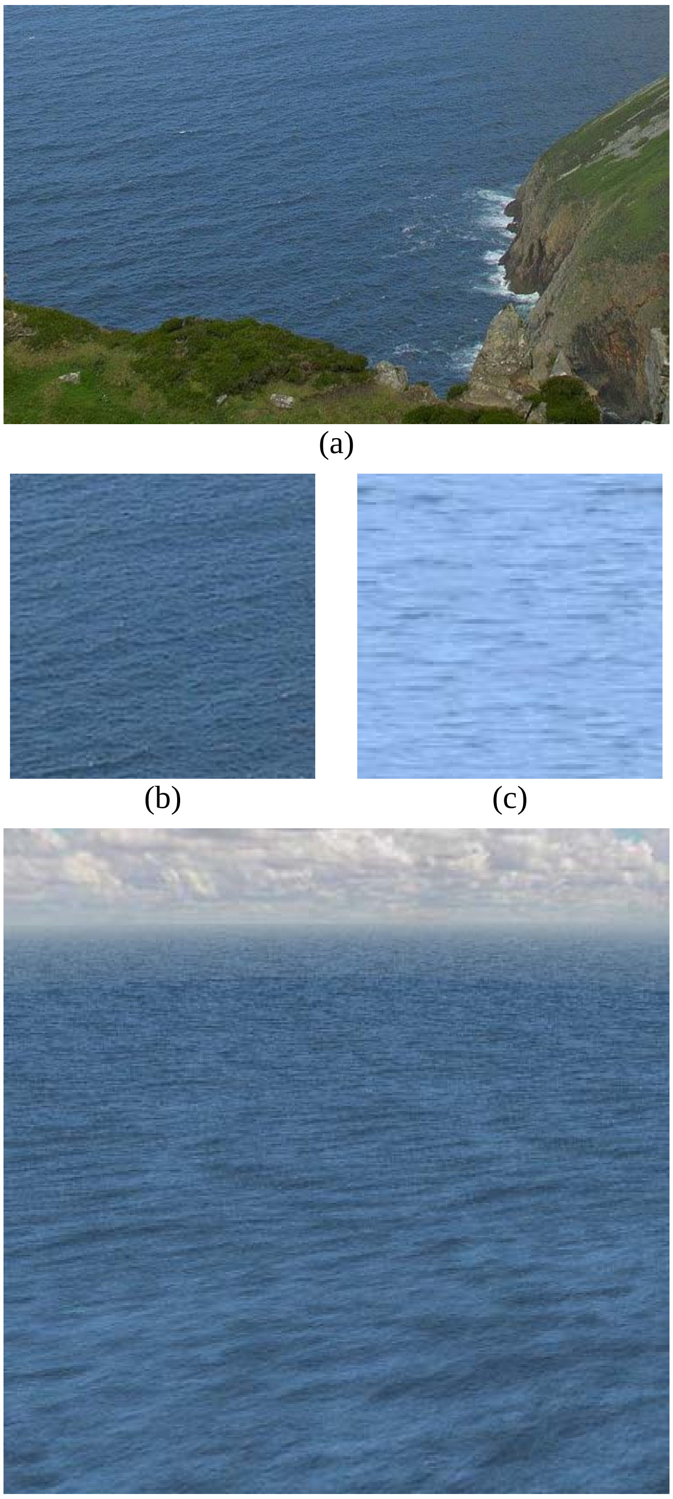


Figure 7. A result for ocean scene.


(a)

(b) (c)

(d)

Figure 8. An example for ocean surface rendering. (a) A real world image borrowed from http://www.public-domain-image.com. (b) The input texture taken from (a). (c) The texture for main waveform. (d) The result for ocean scene.

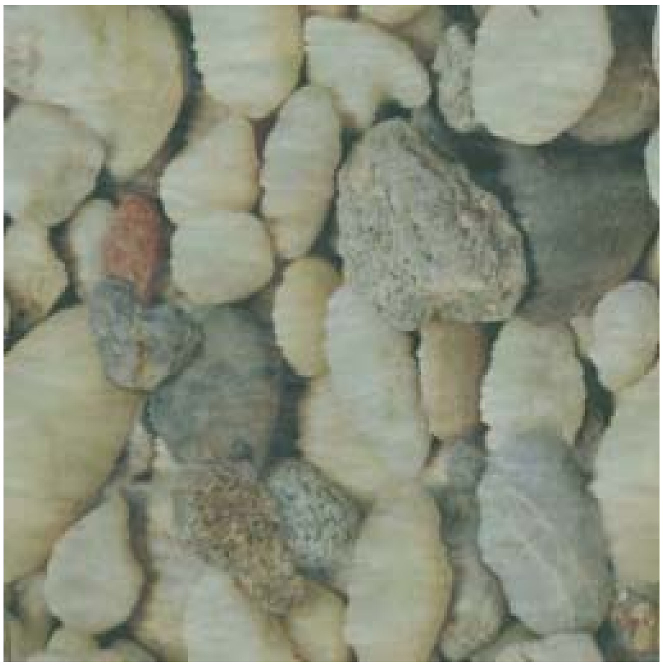


Figure 9. A result for refraction effect.

V. CONCLUSIONS

We present an efficient solution to synthesize an ocean surface using real world textures in this paper. It is convenient to produce ocean scenes by just giving the photo textures. The ocean surface is tessellated straightforwardly with the superposition of the wave height fields with different spatial resolutions. Using the proposed relighting, we can reproduce the wave shading and undulating effects.

However, because the natural texture has its intrinsic color and shading, we can not reproduce complex lighting and stormy effects. In the future, we plan to include the specular effect from a strong light source, more realistic ray-marching effect, and more interactiveness with other objects such as island, shore, and visible seabed.

VI. REFERENCES

[1] K. S. Bhat, S. M. Seitz, J. K. Hodgins, and P. K. Khosla, "Flow-based Video Synthesis and Editing," ACM Transactions on Graphics, Vol. 23, No. 3, pp. 360-363, 2004.

[2] Y. F. Chiu and C. F. Chang, "GPU-based Ocean Rendering, " In Proceedings of IEEE 2006 International Conference on Multimedia and Expo (ICME 2006), pp. 2125-2128, 2006.

[3] E. Darles, B. Crespin, and D. Ghazanfarpour, "Accelerating and Enhancing Rendering of Realistic Ocean Scenes," In Proceedings of WSCG 2007, pp. 287-294, 2007.

[4] H. Fang and J. C. Hart, "Textureshop: Texture Synthesis as a Photograph Editing Tool," ACM Transactions on Graphics, Vol. 23, No. 3, pp. 354-359, 2004.

[5] E. Galin and N. Chiba, "Realistic Water Volumes in Real-Time," In Proceedings of Eurographics Workshop on Natural Phenomena, pp. 1-8, 2006.

[6] D. Hinsinger, F. Neyret, and M. P. Cani, "Interactive Animation of Ocean Waves," In Proceedings of 2002 ACM Siggraph/ Eurographics Symposium on Computer Animation, pp. 161-166, 2002.

[7] Y. Hu, L. Velho, X. Tong, B. Guo, and H. Shum, "Realistic Real-Time Rendering of Ocean Waves," Computer Animation and Virtual Worlds, Vol. 17, No. 1, pp. 59-67, 2006.

[8] E. A. Khan, E. Reinhard, R. W. Fleming, and H. H. Bulthoff, "Image-Based Material Editing," ACM Transactions on Graphics, Vol. 25, No. 3, pp. 654-663, 2006.

[9] J. Kopf, C. W. Fu, D. C. Or, O. Deussen, D. Lischinski, and T. T. Wong, "Solid Texture Synthesis from 2D Exemplars," ACM Transactions on Graphics, Vol. 26, No. 3, Article no. 2, 2007.

[10] V. Kwatra, A. Schodl, I. Essa, G. Turk and A.Bobicks, "Graphcut Textures: Image and Video Synthesis Using Graph Cuts," ACM Transactions on Graphics, Vol. 22, No. 3, pp.277-286, 2003.

[11] V. Kwatra, I. Essa, A. Bobick, and N. Kwatra, "Texture Optimization for Example-Based Synthesis," ACM Transactions on Graphics, Vol. 24, No. 3, pp. 795-802, 2005.

[12] C. H. Lai and J. L. Wu, "Temporal Texture Synthesis by Patch-Based Sampling and Morphing Interpolation," Computer Animation and Virtual Worlds, Vol. 18, No. 4-5, pp. 415-428, 2007.

[13] J. L. Mitchell, "Real-Time Synthesis and Rendering of Ocean Water," ATI Research Technical Report, 2005.

[14] K. Perlin, "Improving Noise," ACM Transactions on Graphics, Vol. 21, No.3, pp. 681-682, 2002.

[15] S. Premoze and M. Ashikhmin, “Rendering Natural Waters,” Computer Graphics Forum, Vol. 20, No. 4, pp. 189-199, 2001.

[16] .A. Schodl, R. Szeliski, D. H. Salesin, and I. Essa, “Video Textures,” In Proceedings of SIGGRAPH 2000, pp. 489-498, 2000.

[17] J. Tessendorf, “Simulating ocean water,” In Simulating Nature: Realistic and Interactive Techniques, ACM SIGGRAPH 2001 course notes No. 47, 2001.

[18] S. Thon and d. Ghazanfarpour, “Ocean Waves Synthesis and Animation using Real World Information,” Computer and Graphics, Vol. 26, No. 1, pp. 99-108, 2002.

[19] L. Y. Wei and M. Levoy, “Fast Texture Synthesis Using Tree-Structured Vector Quantization,” In Proceedings of SIGGRAPH 2000, pp. 479-488, July 2000.

[20] http://en.wikipedia.org/wiki/Graphics_processing_unit