

# Real-time multi-band synthesis of ocean water with new iterative up-sampling technique

E. Miandji · M.H. Sargazi Moghadam ·  
Faramarz F. Samavati · M. Emadi

Published online: 26 March 2009  
© Springer-Verlag 2009

**Abstract** Adapting natural phenomena rendering for real-time applications has become a common practice in computer graphics. We propose a GPU-based multi-band method for optimized synthesis of “far from coast” ocean waves using an empirical Fourier domain model. Instead of performing two independent syntheses for low- and high-band frequencies of ocean waves, we perform only low-band synthesis and employ results to reproduce high frequency details of ocean surface by an optimized iterative up-sampling stage. Our experimental results show that this approach greatly improves the performance of original multi-band synthesis while maintaining image quality.

**Keywords** Real-time rendering · Wave synthesis · Image up-sampling · Programmable graphics hardware

## 1 Introduction

Natural phenomena rendering techniques have become ubiquitous in computer graphics for applications like mo-

tion pictures and games. These rendering techniques usually require complex computations to produce satisfying and physically correct results. Thus, several techniques for alleviation of computational complexity and adaptation of non-real-time methods for real-time applications have been proposed [7, 9, 14, 19]. While the visual quality of produced images is of utmost importance, many performance-intensive applications such as video games demand less load on hardware resources by these rendering methods. That is because, generally, vertex and pixel processors are consumed for rendering of highly detailed meshes along with visual effects such as lighting, shadowing, post-processing effects, etc. Therefore, improving performance of natural phenomena rendering algorithms, which are the most computationally expensive parts of real-time graphics applications such as games, is crucial. This fact also requires maintaining an appropriate balance between quality and performance. We present an optimized approach for multi-band Fourier domain synthesis of “far from coast” deep ocean water, which greatly reduces excessive computational complexity of previous methods and makes it suitable for 3D games. Both synthesis and rendering are performed fully on the GPU to free up CPU for physical computations such as water–boat interactions. Our method first performs a full synthesis for low-band ocean wave frequencies, outputting height map of ocean water at an arbitrary time  $t$ . We use this height map as input for an optimized up-sampling stage to create high frequency height map which is used to produce a normal map for presenting high frequency details of ocean surface. Therefore, in order to reduce processing load, we perform only one synthesis (hence only one FFT [3]), reproducing details by our hand-tuned up-sampling method. In addition, since we are performing low-band synthesis only, this task is done on a smaller data set (e.g. a  $128 \times 128$  texture instead of a  $512 \times 512$  one), which leads to less GPU memory

---

E. Miandji (✉)  
VR Lab, Sharif University of Technology, Tehran, Iran  
e-mail: [E.Miandji@GMail.com](mailto:E.Miandji@GMail.com)

M.H. Sargazi Moghadam · M. Emadi  
Sharif University of Technology, Tehran, Iran

M.H. Sargazi Moghadam  
e-mail: [mhadi\\_sargazim@ee.sharif.edu](mailto:mhadi_sargazim@ee.sharif.edu)

M. Emadi  
e-mail: [m\\_emadi@ee.sharif.edu](mailto:m_emadi@ee.sharif.edu)

F.F. Samavati  
University of Calgary, Calgary, Alberta, Canada  
e-mail: [samavati@ucalgary.ca](mailto:samavati@ucalgary.ca)

bandwidth consumption. Consequently, our primary contributions are to improve rendering performance of previous methods while maintaining visual quality of final results, adapting this natural phenomenon rendering method for game and virtual reality applications even on today's mid-range GPUs.

The remaining of this paper is organized as follows. Section 2 discusses previous works while a brief review of Fourier domain ocean water synthesis [18] and dual-band synthesis method [14] are presented in Sect. 3. This section also discusses the performance of dual-band synthesis approach. Our method is introduced in Sect. 4 followed by hardware implementation details and experimental results of it in Sect. 5. Section 6 provides conclusion and describes future enhancements.

## 2 Related work

Ocean water synthesis methods for computer graphics are categorized in two dominant classes: spatial domain and Fourier domain approaches [14]. Since our method is based on a well-known Fourier domain approach, discussion of spatial domain techniques is excluded from this section.

One of the first computer graphics related elaborative descriptions of water waves was introduced by Fournier and Reeves [6]. They used and modified a well-known ocean wave model, Gerstner waves, to produce water surface waves coming on shoreline; hence simulating shoreline interactions like wave breaking. Many methods have since been presented which use empirical wave models to simulate ocean water. A novel method was presented by Mastin [12], where, given a wind velocity, a Fourier domain filter is created using Pierson–Moskowitz spectrum. A white noise is then filtered to produce a natural ocean scene along with realistic ray-traced lighting. Similarly, Tessendorf [18] used a Fourier domain approach by computing ocean waves based on an empirical wave model, Phillips Spectrum, and a dispersion relation for animating surface waves. He also modified this wind driven wave model for more special wave profiles such as choppy waves. An FFT-based method for generating normal map and a ray-tracing technique for surface wave optics was also proposed. Since Fourier domain approaches are more suitable for real-time applications [19], several new methods have been introduced in order to adapt these approaches for hardware acceleration.

Versatility of Tessendorf's method attracted several researchers to adapt his approach for real-time applications. Jensen and Goliias [7] discussed real-time implementation results of most of Tessendorf's methods, such as wave synthesis (for shallow and deep water), choppy wave profile, foam rendering and surface shading techniques. They also used Navier–Stokes equations for simulating and visualizing small waves caused by floating objects. Lanza

[9] also used Fourier domain synthesis and proposed a novel level-of-detail algorithm based on quad-trees to reduce geometrical complexity and continuous mapping of ocean surface normals. A real-time GPU-based Fourier domain multi-band synthesis which covered a wide range of Tessendorf's work was introduced by Mitchel [14]. Our presented method improves the performance of this technique by introducing an iterative up-sampling stage which enables us to perform only low-band synthesis. In addition to performance improvement, our technique preserves visual quality of Mitchel's method.

## 3 Background

Fourier domain ocean water synthesis approaches are based on statistical models where wave height field is decomposed as a sum of sine and cosine waves. The wave height is considered a random variable of horizontal position and time,  $h(x, t)$  [18]. Also, these methods use analytical semi-empirical models for generation of wave spectrum such as Phillips spectrum and Pierson–Moskowitz spectrum. Tessendorf uses wind driven Phillips spectrum which is very versatile for synthesizing deep ocean waves and waves that appear to be approaching a shoreline:

$$P_h(K) = A \frac{e^{-1/(kL)^2}}{k^4} |\hat{K} \cdot \hat{\omega}|^2$$

In this equation,  $L = V^2/g$  is the largest possible wave arising from a continuous wind of speed  $V$ ,  $g$  is the gravitational constant,  $K$  is wave vector with components  $K_x = 2\pi n/L_x$  and  $K_y = 2\pi m/L_y$ ,  $L_x$  and  $L_y$  are horizontal and vertical length of water patch,  $n$  and  $m$  are integers with boundaries  $-M/2 \leq m < M/2$  and  $-N/2 \leq n < N/2$ ,  $N$  and  $M$  are the number of horizontal and vertical height map grid points,  $\omega$  is the direction of the wind and  $A$  is a numeric constant. Moreover,  $K$ , the wave vector, and  $k = 2\pi/\lambda$ , the wave number, have the following relation:  $k = |K|$ . As Tessendorf states, the cosine factor  $|\hat{K} \cdot \hat{\omega}|^2$  in the Phillips spectrum eliminates waves that move perpendicular to the wind direction and the exponent in the denominator affects directionality of waves.

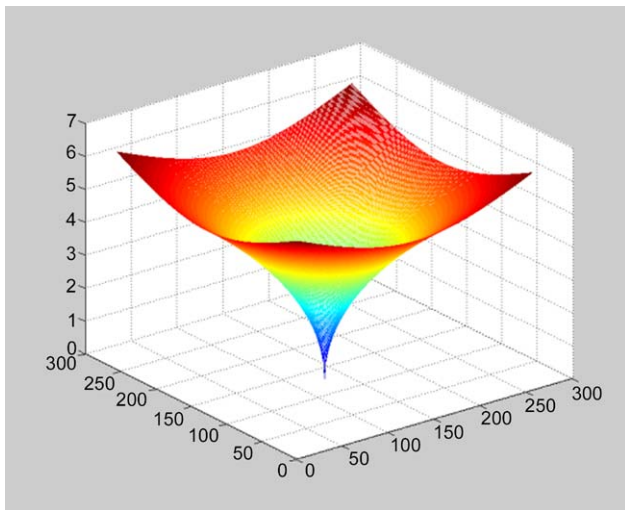
Using Phillips spectrum, complex Fourier amplitudes at time zero are calculated by the following equation:

$$\hat{h}_0(K) = \frac{1}{\sqrt{2}} (\varepsilon_r + i\varepsilon_i) \sqrt{P_h(K)}$$

where  $\varepsilon_r$  and  $\varepsilon_i$  are ordinary independent draws from a Gaussian random number generator, with mean 0 and standard deviation 1. These numbers tend to follow the experimental data on ocean waves. Given the dispersion relation  $\omega(K) = \sqrt{gk}$  for deep-water wave animation (Fig. 1),

**Table 1** Performance comparison between low-band and high-band syntheses. These values are extracted from our reference implementation of Mitchel’s method (values specified are only GPU frame times)

Operation	Low-band synthesis (128 × 128)		High-band synthesis (512 × 512)	
	Time (ms)	Inst. slots	Time (ms)	Inst. slots
Computing Fourier amplitudes	0.165	25	2.915	25
Horizontal Scramble (FFT)	0.217	4	6.525	4
Horizontal Butterflies (FFT)	0.27 * 7	18	3.85 * 9	18
Vertical Scramble (FFT)	0.129	6	3.85	6
Vertical Butterflies (FFT)	0.27 * 7	18	3.85 * 9	18
Scaling	0.048	4	0.571	4
Normal Map Generation	Not required	Not required	2.267	44



**Fig. 1** Angular frequencies using dispersion relation shown as a mesh

Fourier amplitude of the wave field at time  $t$  is calculated by:

$$\hat{h}(K, t) = \hat{h}_0(K)e^{i\omega(k)t} + \hat{h}_0^*(-K)e^{-i\omega(k)t} \tag{1}$$

The final result of synthesis, which is the height map at time  $t$ , is computed by inverse Fourier transform (2) of function  $\hat{h}(K, t)$ . Therefore,  $h(K, t)$  represents the height value of computed height field at position  $X = (x, y)$  and time  $t$  as a sum of sinusoids with complex time-dependent amplitudes:

$$h(X, t) = \sum_K \hat{h}(K, t)e^{iK.X} \tag{2}$$

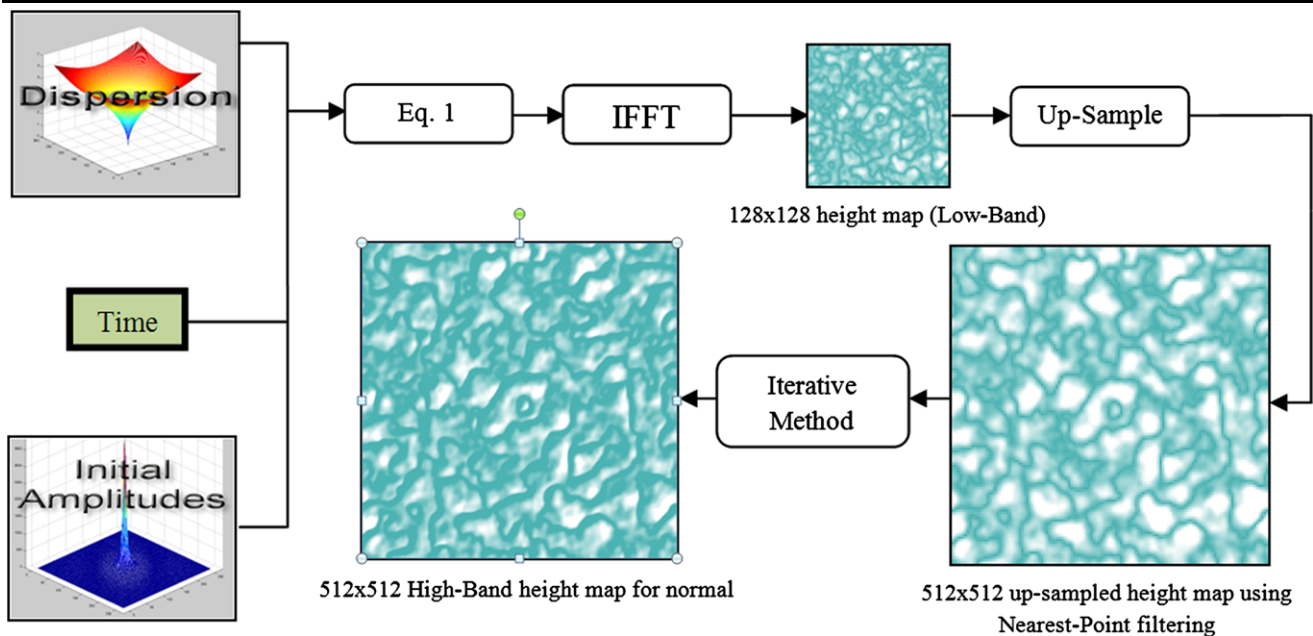
This synthesized height map will be used to displace the even-spaced grid patch of water using vertex texturing capability of shader model 3.0 hardware or above (similar methods may be used: such as Render to Vertex Buffer—R2VB). Unfortunately, since vertex texturing is computationally expensive for hardware, constraints should be placed for number of vertex texture fetches per frame; hence limiting water patch size to fairly small sizes for real-time applications.

Mitchel [14] presented a method to overcome this limitation by introducing dual-band synthesis. In this technique, a low-band synthesis is used to compute a low frequency height map for displacement mapping while a high-band synthesis is employed to produce a high frequency height map which will be used for generating a normal map for bump mapping. Since the low frequency height map has fewer pixels than the high frequency one, fewer vertex texture fetches will be required. Each of these syntheses consists of three main steps (excluding pre-process computations): Computation of Fourier domain amplitudes, hardware accelerated Inverse FFT and generation of normal map from height map (low-band synthesis does not require the last step). Table 1 illustrates time consumed per frame for each subsection of synthesis states. It is evident that high frequency wave synthesis consumes considerably more processing power of GPU than low-band synthesis (approximately 20 times). Thus, we used this fact as our motivation for optimizing multi-band synthesis of ocean waves by performing only low-band synthesis and producing high frequency details by an iterative up-sampling stage. This stage receives results of low-band synthesis and iteratively adds high frequency details in order to produce high frequency height map for normal mapping.

### 4 Method

Ocean water simulation and visualization can be broken into two major parts: wave synthesis and rendering. While the former is responsible for generating height map, the latter uses results of the first part to displace water patch surface and add lighting effects.

Several level-of-detail techniques have been proposed for ocean surface rendering [8] since ocean surface holds several properties of terrain except that the displacement data is static. Nonetheless, view-dependent level-of-detail terrain rendering techniques, like quad-trees, geometry clip maps, etc., are eligible to be used for large-scale ocean water rendering.



**Fig. 2** Multi-band synthesis of ocean waves using our introduced method

Unlike rendering optimization techniques, less work has been done to increase performance of GPU-based synthesis. Given the fact that synthesis consumes more than half of a typical frame time (based on our experimental results and previous implementations [14]), it can be inferred that to obtain promising visual quality while maintaining stability and performance it is substantial to optimize GPU-based synthesis. This optimization can be useful in two ways:

- Saved processing power can be utilized to add more sophisticated wave profiles (e.g. choppy waves) or lighting-shading techniques.
- Higher frame rates can be achieved, which is critical for multi-player games.

Owing to the fact that dual-band synthesis along with a proper level-of-detail algorithm can greatly reduce rendering time, our method focuses on optimization of dual-band synthesis to adapt this technique for games. To achieve this, a number of techniques can be used:

- Reducing height field size and increasing tiling: although the fastest to implement, this technique suffers from ocean surface repetitiveness and will diminish visual quality, especially if seen from higher altitudes.
- Because high-band synthesis results in a high frequency height map, it is possible to downsample and apply a low-pass filter to this height map in order to obtain low frequency height values [14]. Although, comparing to first option, this technique would significantly increase image quality, it has two notable drawbacks: 1. An ordinary GPU-friendly down-sample operation accompanied

with a low-pass filter may degrade final height values of the height field. Furthermore, since low-band height map is used for displacement of water surface and our wave model is semi-empirical, assorted defections will occur in scenarios such as simulation applications. For instance, a military computer-based training program for boat simulation may suffer from poor interactive results because of attenuation of large waves. 2. Even if an ideal filter is designed, since the high-band synthesis is intended, our computations have to be done on a remarkably larger data which, as illustrated in Table 1, dissipates approximately twenty times more of frame time on a typical synthesis scenario and a mid-range consumer GPU.

We introduce an approach which overcomes limitations and defects of aforementioned methods. A low-band synthesis is first performed, outputting height field for displacement mapping (since we are dealing with low frequencies, this step is much computationally cheaper than the high-band synthesis). Following low-band synthesis, low frequency height field is used as input for a resolution enhancement and high frequency detail reproduction stage (Fig. 2). To achieve both of these goals, we use an iterative high frequency preserving up-sampling method. Our experimental results show that this approach not only performs significantly faster, but produces more visually satisfactory results since our up-sampling method adds extra edges on height map which amplifies choppy of waves. Moreover, since the virtual high-band height map is used for generating surface normals, our method does not affect height values used for displacement mapping; therefore making it suitable for physically correct simulations used for military applications.

### 4.1 Up-sampling

In digital image processing, low resolution images are widely used because of their smaller size, which results in less processing. These images are then adapted to the screen size via several techniques.

This wide use in multimedia applications has made up-sampling a challenging area both in image quality and real-time implementation; thus attracting much attention from researchers and system developers.

Re-sampling or up-sampling can be performed in either software or hardware. Software solution has the advantages of ability to process large amounts of images and yet flexibility of operating on the entire image line data. However, great disadvantage of these methods is that they do not produce images in real time, which makes them almost impossible to be used in applications like video games and environment simulations. In contrast, hardware-based techniques produce images in real time while there is a limited processor and memory space available. Therefore, methods to implement up-sampling should take into account this lack of processor power and memory. A primary technique is inserting pixels with the same value as adjacent pixels which maintain high frequency content while adding a known image artifact of “waxing and waning.” Bilinear interpolation technique uses a weighted averaging between two nearest input pixels, which significantly causes the loss of high frequency contents of the image and thus “blurring” of the image.

Cubic convolution and cubic spline interpolation are other techniques that eliminate high frequency contents of the image [13, 16].

Practical image up-sampling methods use filtering to remove high frequency distortions derived from padding. As mentioned, these filters cause image blurring and so our goal to construct high detail sets through up-sampling cannot be achieved. Image deblurring is one solution that can be used to improve the quality of a blurred image. Blurring-and-deblurring model can be demonstrated as in Fig. 4.

Blurring model can be expressed as:

$$g(x, y) = f(x, y) * h(x, y) + n(x, y)$$

where  $g(x, y)$  is blurred image,  $f(x, y)$  is input image for blur function  $h(x, y)$  and  $n(x, y)$  is the noise function. Therefore, having blurred image  $g(x, y)$ , deblurring problem deals with finding a filter for constructing the best estimate  $\hat{f}(u, v)$ . If we rewrite the above equation in frequency domain, we have:

$$G(u, v) = F(u, v)H(u, v) + N(u, v)$$

Methods for regularized pseudo-inverse calculation of  $H$ , such as the Tikhonov, Truncated Singular Value Decomposition (TSVD) and Wiener filters, attempt to estimate the Point

Spread Function (PSF) from the image statistics and use regularization to compute the inverse kernel [2, 17]. For example, the classical Wiener filter computes the inverse matrix

$$T(u, v) = \frac{H^*(u, v)}{|H(u, v)|^2 + \frac{S_n(u, v)}{S_f(u, v)}}$$

where  $S_n(u, v)$  and  $S_f(u, v)$  are the power spectra of the noise and the original image, respectively. Other methods, such as Constrained Least Squares Filtering, seek to constrain the variation in the image due to the noise without actual knowledge of either power spectrum. This method can be stated as

$$\hat{F}(u, v) = \left[ \frac{H^*(u, v)}{|H(u, v)|^2 + \gamma|P(u, v)|^2} \right] G(u, v)$$

where  $P(u, v)$  is the Fourier transform of a  $3 \times 3$  Laplacian filter and  $\gamma$  is a constant value. Computer vision methods such as graph cuts and belief propagation have also been used to recover nearly optimal deblurred images [4]. All these methods depend on the previous knowledge of the transfer function of the blurring  $H(u, v)$ . If the blur function is not known, then recovery of  $F(u, v)$  from  $G(u, v)$  is more difficult. So extraction of PSF of blurring model ( $H$ ) is not always possible.

Now consider a  $128 \times 128$  image as low detail data set which should be up-sampled to achieve a  $512 \times 512$  high detail image. First step is a one-to-one mapping of low detail image pixels to make a  $512 \times 512$  primary image. This is the same as pixel inserting technique mentioned previously in this section. Now, considering the original high detail  $512 \times 512$  image which should be estimated, we can see that extraction of the primary image from this high detail image cannot be modeled as a single shift invariant PSF [1]. As mentioned earlier, deconvolution methods cannot be used to extract an estimate for the high detail image from the primary image.

Blind deconvolution is a description of a class of techniques that estimate  $H(u, v)$  from one or more images and then use that estimate to recover an estimate of  $F(u, v)$ . Existing blind deconvolution methods typically assume that the blur kernel has a simple parametric form, such as Gaussian or low-frequency Fourier components [5, 10]. However, as such assumptions can prevent high frequencies (such as edges) from appearing in the reconstruction, in this application we cannot use these methods. On the other hand, these methods require much computational cost. Here we have introduced a method which extracts an estimate of the original high detail image using an iteration which is efficient in both computational complexity and quality.

### 4.2 Presented iterative method

Our main purpose is to find an estimate for the output image. Imagine an unknown function  $H$  which constructs the pri-

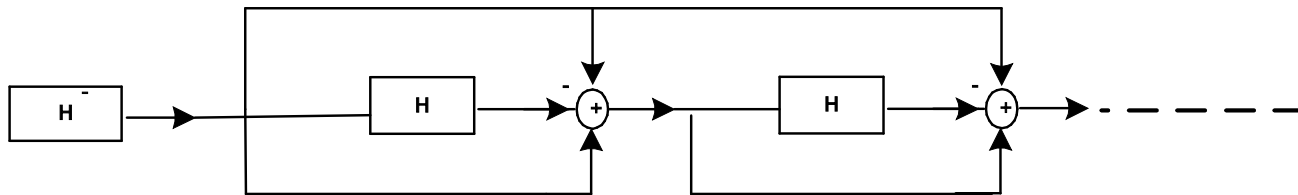


Fig. 3 Implementation block diagram of iterative method presented in (5)

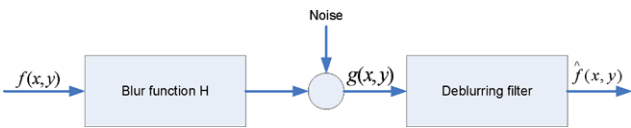


Fig. 4 Blurring-and-deblurring model

mary image mentioned above from the original high detail image; our goal is to find the output of  $H^{-1}$ .

If we define  $I$  as the unit operator then:

$$H^{-1} = \frac{I}{H} = \frac{I}{I - E} \tag{3}$$

In this equation we assume that  $E$  is the process error or operator error. By using Taylor expansion for this function we have:

$$H^{-1} = \frac{I}{I - E} = I + E + E^2 + E^3 + E^4 + \dots$$

The outputs in  $i$ th and  $(i + 1)$ th iterations are as follows:

$$\begin{aligned} \hat{F}^{(i)} &= (I + E + E^2 + \dots + E^{i-1})HF \\ \hat{F}^{(i+1)} &= (I + E + E^2 + \dots + E^i)HF \end{aligned} \tag{4}$$

Consider that  $F$  is the desired high resolution image. Therefore, according to the above equations, it could be easily shown that:

$$\hat{F}^{(i+1)} = (I - H)\hat{F}^{(i)} + HF \tag{5}$$

For the convergence of the algorithm we must have:

$$(e^{(i+1)} = \hat{F}^{(i+1)} - F) < (e^{(i)} = \hat{F}^{(i)} - F)$$

By using (4), one can easily show that the error in the  $i$ th iteration is equal to:

$$e^{(i)} = -E^i F \tag{6}$$

Thus the algorithm is converged if and only if  $\|E\| < 1$ . Therefore, it is obvious that the smaller  $E$  results in faster convergence. Also, it is apparent that when  $E$  is not less than unity, the convergence of the iterations is not guaranteed. We can use a relaxation factor named  $\lambda$  to have a faster convergence ( $\lambda > 1$ ). Hence (5) converts to:

$$\hat{F}^{(i+1)} = (I - \lambda H)\hat{F}^{(i)} + \lambda HF$$

This iterative approach is widely used in several distortion compensation systems [11]. The block diagram of this method is shown in Fig. 3.

Although this method shows significant quality, it requires several iterations for convergence. Therefore we have modified this method to a method which will have faster convergence and stability. If we use another expansion in (3), we have:

$$\begin{aligned} H^{-1} &= \frac{I}{I - E} = \frac{I + E}{I - E^2} \\ &= (2I - H)(I + E^2 + E^4 + E^6 + \dots) \end{aligned}$$

The outputs in the  $i$ th and  $(i + 1)$ th iterations will be:

$$\begin{aligned} \hat{F}^{(i)} &= (2I - H)(I + E^2 + E^4 + \dots + E^{2(i-1)})HF \\ \hat{F}^{(i+1)} &= (2I - H)(I + E^2 + E^4 + \dots + E^{2i})HF \end{aligned}$$

According to the above equations it could be simply shown that

$$\begin{aligned} \hat{F}^{(i+1)} &= \hat{F}^{(i)} + H(H\hat{F}^{(i)}) \\ &\quad - 2H\hat{F}^{(i)} + 2HF - HHF \end{aligned} \tag{7}$$

By using (7) one can easily show that the error in the  $i$ th iteration is equal to:

$$e^{(i)} = -E^{2i} F \tag{8}$$

The block diagram of the method is shown in Fig. 5.

As a result of presented discussion, it can be inferred that the iterative performance depends directly on the election of a proper iteration function  $H$ . We have used a simple average filter with two taps for up-sampling with factor 4. Results show that this function produces sufficient detail and quality for our application.

Comparing (6) and (8) it is evident that the modified method is faster than the previous method in number of iterations at the expense of using more computational cost; however, comparing the results from both methods shows that the convergence is much faster in the modified method and that a total computation cost is reduced. Noting the experimental results, the convergence of this method could be

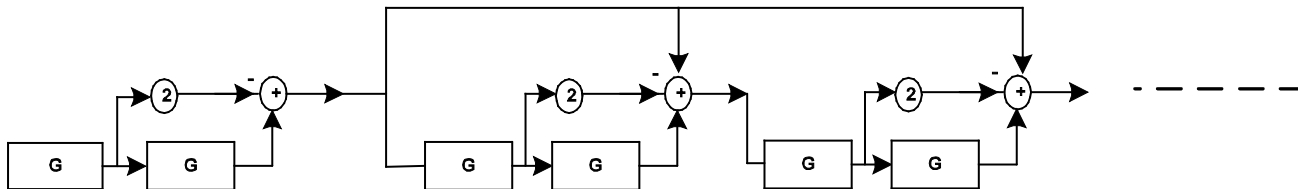


Fig. 5 Implementation block diagram of presented iterative method illustrated in (7)

profoundly improved using weighting functions in the iteration sequence (e.g. Chebyshev weighting). In our implementation, we have constructed an experimental weight function which shows best results for an ocean scene rendered in real time with moderate wind speed.

### 5 Implementation and results

We implemented and tested our approach using shader model 2.0 and DirectX 9.0 on an NVIDIA GeForce 7600Go. Both the normal multi-band synthesis method presented by Mitchel [14] and our approach were implemented to compare the results. The FFT was implemented employing the method presented by Mitchel et al. [15] and data precision for all stages of synthesis was 32-bit single channel and 64-bit dual channel floating point textures using Direct3D’s D3DFMT\_R32F and D3DFMT\_G32R32F texture formats, respectively. Our modified iterative method, as seen in Fig. 5, uses two-stage filtering with a 2-tap filter. Comparing experimental results, a weight function equal to [0.55, 0.6, 0.9, 0.1, 0.8] was applied to the iteration function. Results for 1, 3 and 5 iterations are demonstrated in Fig. 7. As shown, five iterations are adequate to achieve proper detail in the high-band image.

Deploying a typical test configuration,  $128 \times 128$  and  $512 \times 512$  image size for low-band and high-band syntheses, respectively, we were able to achieve a better performance comparing to the implementation without our optimization method. Figure 6 illustrates performance of our method comparing to non-optimized method using different test configurations for synthesis detail in three different image resolutions. Moreover, scene rendering was turned off for both our approach and the previous method for a better comparison of synthesis results.

Our implementation of ocean surface rendering consists of a simple Phong lighting, reflection, refraction, Fresnel effect, depth simulation, along with a sky dome and a small underwater terrain. The results obtained from our optimized synthesis method show that the image quality of our method is almost identical to that of a not-optimized dual-band synthesis approach, except for a little degradation of high-band synthesized height map which in a typical scene is not notable to human eye. Additionally, since high-band height

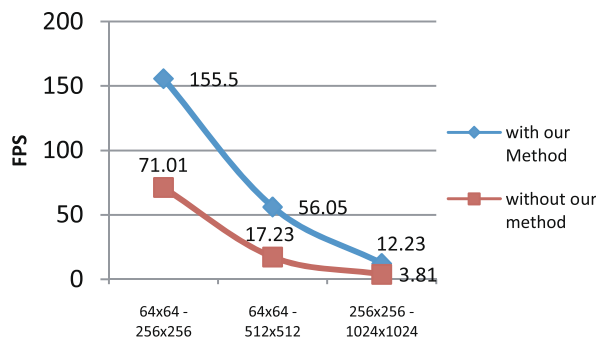


Fig. 6 Performance results for different test scenarios

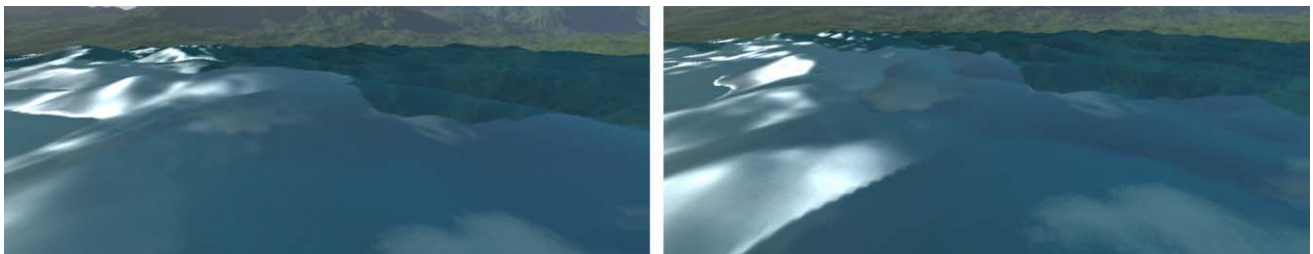
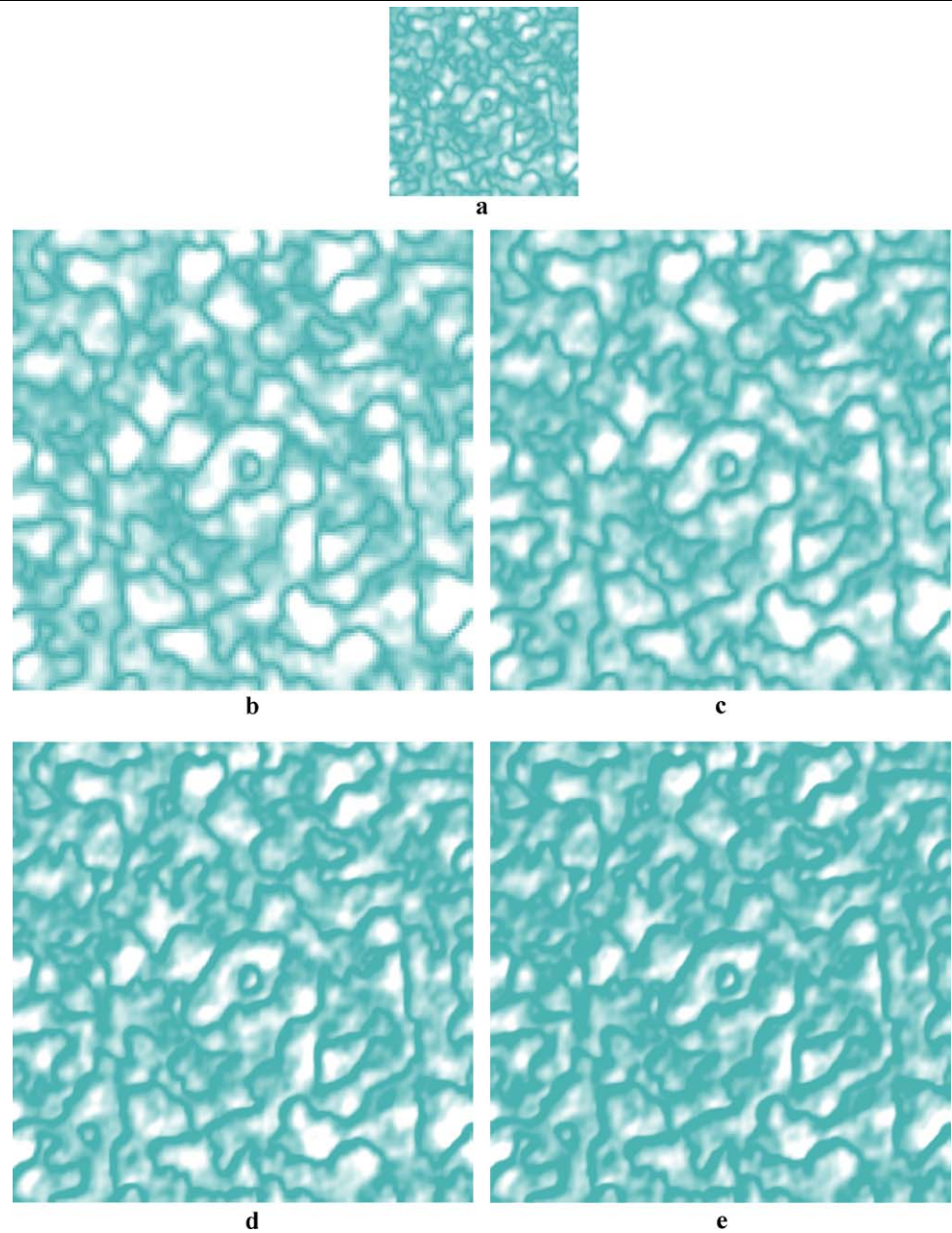
map is used for generating normal map, this issue does not affect the quality and realism of final produced images. Figure 8 compares a typical ocean scene rendered with our optimized approach and that without it.

### 6 Conclusion and future work

We presented a simple, yet effective, method for optimization of multi-band Fourier domain synthesis of ocean waves. Our approach reduces computational cost of multi-band synthesis by performing only one low-band synthesis and reproducing high frequency details by using our iterative up-sampling algorithm. Our experimental results show that the proposed method is highly efficient comparing to the original method. Additionally, results of implementation show that our method preserves image quality of original method.

Although our technique greatly reduces time complexity and enables utilization of ocean wave synthesis in real-world games even on mid-range GPUs, much work can be done to enhance performance further. One method would be to perform synthesis once every two frames and averaging synthesis results (height map) to construct the intermediate frame’s height map. This technique can be extended for a better performance but more implementation complexity; for instance, creating height map every three frames by distributing synthesis steps among frames and employing Multiple Render Target (MRT) capability of hardware. These techniques usually unbalance frames creating non-smooth animation and unsteady frame rates. Therefore, a proper

**Fig. 7** Demonstration of our modified iterative method with weights [0.55, 0.6, 0.9, 0.1, 0.8] for up-sampling a  $128 \times 128$  height map to a  $512 \times 512$  high detail image. **(a)** Original  $128 \times 128$  height map image as input of up-sampling stage. **(b)** Output of hardware nearest-point sampling; initial  $512 \times 512$  image for iteration. **(c)** First step of iteration resolves “waxing and waning” artifact yet the image is blurred. **(d)** Third step of iteration enhances image sharpness but it is not yet proper for high frequency height map. **(e)** The fifth iteration produces the high detail height map although more iteration can be added for even better quality



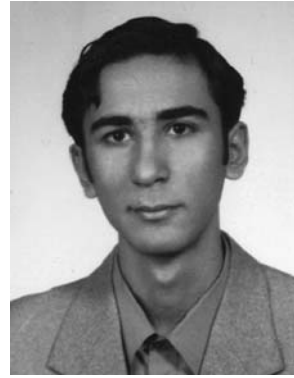
**Fig. 8** Comparison of final rendered image using our method (*left*) and a previous un-optimized method (*right*)

method which utilizes this concept and preserves steadiness of frame rates can further improve the synthesis performance.

Apart from wave synthesis optimization techniques, several rendering and shading optimization techniques can be employed. An efficient adaptive view-dependent triangulation algorithm along with our multi-band synthesis method can greatly reduce rendering time since, as mentioned earlier, vertex texture fetch is currently expensive on hardware.

## References

1. Ben-Ezra, M., Lin, Z., Wilburn, B.: Penrose pixels; super-resolution in the detector layout domain. In: IEEE International Conference on Computer Vision (ICCV) (2007)
2. Chan, T.F., Shen, J.: Theory and computation of variational image deblurring. In: Mathematics and Computation in Imaging Science and Information Processing. IMS (Inst. Math. Sci.) Lecture Notes Series. World Scientific, Singapore (2007)
3. Cooley, J., Tukey, J.: An algorithm for the machine calculation of complex Fourier series. *Math. Comput.* **19**, 297–301 (1965)
4. Feng, L., Jingyi, Y., Jinxiang, C.: A hybrid camera for motion deblurring and depth map super-resolution. In: CVPR 2008 (2008)
5. Fergus, R., Singh, B., Hertzmann, A., Roweis, S.T., Freeman, W.T.: Removing camera shake from a single photograph. *ACM Trans. Graph.* **25**(3), 787–794 (2006) (Proc. SIGGRAPH 2006)
6. Fournier, A., Reeves, W.-T.: A simple model of ocean waves. *Comput. Graph.* **20**, 75–84 (1986)
7. Jensen, L.-S., Goliás, R.: Deep-water animation and rendering. In: Game Developers Conference Europe (2001)
8. Kryachko, Y.: Using Vertex Texture Displacement for Realistic Water Rendering. *GPU Gems II*. Addison-Wesley, Reading (2005)
9. Lanza, S.: Animation and Display of Water. *ShaderX3: Advanced Rendering with DirectX and OpenGL*. Charles River Media, Hingham (2004)
10. Levin, A.: Blind motion deblurring using image statistics. In: Advances in Neural Information Processing Systems (NIPS), Dec. 2006
11. Marvasti, F.A.: An iterative method to compensate for the interpolation distortion. *IEEE Trans. ASSP* **37**(1), 1617–1621 (1989)
12. Mastin, G.-A., Watterger, P.-A., Mareda, J.-F.: Fourier synthesis of ocean scenes. *IEEE Comput. Graph. Appl.* **April/March**, 16–23 (1987)
13. Messing, K.S., Ibrahim, S.: Image up-sampling technique. European Patent 1947603 A2 (2008)
14. Mitchell, J.-L.: Real-time synthesis and rendering of ocean water. ATI Research Technical Report, Marlboro, MA (2005)
15. Mitchell, J.-L., Ansari, M.-Y., Hart, E.: Advanced Image Processing with DirectX 9 Pixel Shaders. *ShaderX 2: Shader Programming Tips & Tricks with DirectX 9*. Wordware, Texas (2004)
16. Pathak, B.: Method to up-sample frequency rich images without significant loss of image sharpness. US Patent 6747630 B2 (2004)
17. Piotr, W., Nagy, G.: Large scale image deblurring in Java. In: Proc. of ICCS, pp. 721–730 (2008)
18. Tessendorf, J.: Simulating ocean water. *Simulating nature: realistic and interactive techniques course notes*. In: SIGGRAPH 1999 (1999)
19. Thon, S., Dischler, J.-M., Ghazanfarpour, D.: Ocean waves synthesis using a spectrum-based turbulence function. In: Proceedings of the International Conference on Computer Graphics (2000)



**E. Miandji** received his B.Sc. in Software Engineering from Azad University of Tehran, Iran in 2008. He is currently a researcher and lead programmer at “Shahid Rezayi Research Center–Sharif University of Technology”. He is also a member in “VR Lab–Sharif University of Technology”. His main research interests are real-time rendering techniques, computer vision and real-time environment simulation. He is also an experienced C++ and shader programmer.



**M.H. Sargazi Moghadam** received his B.Sc. in Electrical Engineering from Sharif University of Technology in 2006. He is now studying his M.Sc. degree in Communications Engineering at Sharif University of Technology, Tehran, Iran. His main research interests are digital signal processing, wireless communications systems, coding and cryptography. He has published two IEEE conference papers and has submitted one PIERs journal paper.



**Faramarz F. Samavati** is an Associate Professor in the Department of Computer Science, University of Calgary. He received his Ph.D. degree from Sharif University of Technology in 1999. Prof. Samavati’s research interests are computer graphics, geometric modeling, 3D imaging, and interactive modeling. He has authored more than 50 research papers in subdivision surfaces, sketch-based modeling, multiresolution and wavelets, surface modeling and scientific visualization.



**M. Emadi** received the B.Sc. and M.Sc. degrees in Electrical Engineering from Tehran and Sharif Universities of Technology in 2004 and 2006, respectively. He was with the VLSI and SRRC lab from 2004 to 2007, also he was a visiting lecturer at Azad University and he taught some courses at Sharif University of technology. He is now the Ph.D. student and Manager of an Advanced Research Center in Sharif University of Technology. His research interests are in digital and radar signal processing and its applications. He

was honored at IEEE paper contest in Iran and was known as an exceptional talent in the Ministry of Science.