Graphics processing unit accelerated computation of digital holograms

Hoonjong Kang,* Fahri Yaraş, and Levent Onural

Department of Electrical and Electronics Engineering, Bilkent University, TR -06800 Bilkent, Ankara, Turkey

*Corresponding author: hjkang@ee.bilkent.edu.tr

Received 6 July 2009; revised 26 September 2009; accepted 28 September 2009; posted 29 September 2009 (Doc. ID 113855); published 12 October 2009

An approximation for fast digital hologram generation is implemented on a central processing unit (CPU), a graphics processing unit (GPU), and a multi-GPU computational platform. The computational performance of the method on each platform is measured and compared. The computational speed on the GPU platform is much faster than on a CPU, and the algorithm could be further accelerated on a multi-GPU platform. In addition, the accuracy of the algorithm for single- and double-precision arithmetic is evaluated. The quality of the reconstruction from the algorithm using single-precision arithmetic is comparable with the quality from the double-precision arithmetic, and thus the implementation using single-precision arithmetic on a multi-GPU platform can be used for holographic video displays.

OCIS codes: 090.1760, 090.2870.

1. Introduction

Various fast digital hologram generation algorithms have been reported in the literature [1–6]. One such algorithm is the coherent holographic stereogram (CHS) method, which is an approximation. This method is based on partitioning the hologram plane. Digital holograms can be generated by the CHS method at a much faster speed than that of conventional approaches [7]. However, the quality of the phase-added stereogram [7], which is the original version of the CHS, is not sufficient for many applications as a consequence of the approximations used. To improve the quality, improved and modified versions have been reported [8–12]. The latest version is the accurate compensated phase-added stereogram (ACPAS) [12], and it has significant advantages such as high quality reconstruction and fast generation. Therefore, the ACPAS can be used for real-time electroholographic displays.

Field-programmable gate arrays (FPGAs) or graphics processing units (GPUs) are often used for fast implementation of highly complex algorithms [1,13,14]. FPGA based approaches have some drawbacks such as the expense, long development time, and the need for more specialized technical know how. On the other hand, an accelerated computing system based GPU could have many advantages such as high computational power, low cost of the GPU board, flexibility in algorithm modification, flexibility in hardware extension, and a short development time. Therefore, the approximated digital hologram generation algorithm, ACPAS, is implemented on a GPU platform.

Here we implement the ACPAS on a multi-GPU platform with three GPUs. The performance of the ACPAS on the GPU implementation is assessed and compared with a typical CPU implementation that is based on a source model that assumes that each object point radiates as an optical point source. We refer to such a source model based hologram computation result as a reference hologram. In addition, we evaluate the accuracy of the ACPAS computed by a single precision floating point data format on a GPU.

0003-6935/09/34H137-07$15.00/0

2. Fast Digital Hologram Generation

A. Coherent Stereogram Calculation

The CHS is a complex-reduction approximation used for digital hologram generation, and the ACPAS [12,15] is the latest improved version of the CHS. Reconstruction from the ACPAS is almost the same as the reconstruction from the corresponding reference hologram [15], and the computational speed of the ACPAS is much faster because of the reduced computational complexity. The main reason for the faster computation time is the initial partitioning of the output holographic fringe pattern into segments. In the case of reference hologram, each object point contributes a zone-plate term, and the hologram is a superposition of such terms. In the case of the CHS, each segment has a single spatial frequency component that corresponds to each object point as a consequence of the adopted approximation. Therefore, the generated fringe pattern over a segment becomes a superposition of complex sinusoids. To achieve more accurate beam steering, the ACPAS uses two kinds of improvement: phase compensation [10] and finer discretization at the spatial frequency domain [12].

The complex form of the ACPAS in hologram plane (ξ,η), using a point cloud in the (x,y,z) volume, is computed as [10]

$$\mathbf{I}_{\mathrm{ACPAS}}(\xi,\eta)=\sum_{p=1}^{N}\frac{a_p}{r_p}\exp\{j2\pi[(\xi-\xi_c)f_{p\xi c\mathrm{int}}+(\eta-\eta_c)f_{p\eta c\mathrm{int}}]+jkr_p+j\phi_p+jC_{\xi\eta}\},\tag{1}$$

where N is the number of object points, and a_p and ϕ_p are the amplitude and the phase of an object point, respectively. Wavenumber k is $2\pi/\lambda$, where λ is the free-space wavelength of the coherent light. The distance r_p between the $p-th$ object point and point (ξ_c,η_c) on the hologram is $[(\xi_c-x_p)^2+(\eta_c-y_p)^2+z_p^2]^{1/2}$. The discrete spatial frequencies $f_{p\xi c\mathrm{int}}$ and $f_{p\eta c\mathrm{int}}$, whose units are both cycles per unit length, are determined as $f_{p\xi c\mathrm{int}}=\xi_{\mathrm{int}}f_{\mathrm{FSF}}$ and $f_{p\eta c\mathrm{int}}=\eta_{\mathrm{int}}f_{\mathrm{FSF}}$, where the fundamental spatial frequency (FSF) component f_{FSF} is the reciprocal of the segmentation size. The related spatial frequency error compensations $C_{\xi\eta}$, in radians per unit length, are determined as

$$C_{\xi\eta}=2\pi[(f_{p\xi c}-f_{p\xi c\mathrm{int}})(\xi_c-x_p)+(f_{p\eta c}-f_{p\eta c\mathrm{int}})(\eta_c-y_p)],\tag{2}$$

where $f_{p\xi c}$ and $f_{p\eta c}$ are the continuous spatial frequencies on the ξ and η axes, respectively. Spatial frequencies $f_{p\xi c}$ and $f_{p\eta c}$ are determined as

$$f_{p\xi c}=(\sin\theta_{p\xi c}-\sin\theta_{\xi\mathrm{ref}})/\lambda,\tag{3}$$

$$f_{p\eta c}=(\sin\theta_{p\eta c}-\sin\theta_{\eta\mathrm{ref}})/\lambda,\tag{4}$$

where $\theta_{p\xi c}$ and $\theta_{\xi\mathrm{ref}}$ are the incident angles of the object and reference beams on the ξ axis, and $\theta_{p\eta c}$ and $\theta_{\eta\mathrm{ref}}$ are the incident angles on the η axis.

The computation of the ACPAS for each segment using inverse fast Fourier transform (IFFT) comprises two steps: calculation of the contribution from each object point and an IFFT. In the first step, the output hologram plane is divided into suitable square segments. The spatial frequency, phase, and phase compensation that corresponds to each object point are determined for each segment, and this yields the coefficients that correspond to the 2D complex sinusoids at the discrete Fourier domain. These coefficients form the spectrum of the signal over one segment. In the second step, each spectrum associated with a segment is transformed by the IFFT. The computation of the ACPAS is completed by repeating this procedure for each segment.

B. Computational Complexity

Since the contribution of each object point that corresponds to each pixel on the hologram plane is calculated and recorded eventually as a digital hologram, the worst-case performance of the direct reference hologram computation algorithm is $O(nm)$, where n is the number of pixels of the digital hologram and m is the number of object points. On the other hand, in the case of the CHS, the contribution of each object point is calculated only to the center pixel on each segment, and thus the computational complexity is significantly reduced. Therefore, the complexity of the CHS algorithm is $O(ms)$, where s represents the number of segments. Even though there are additional IFFT operations, their contribution to the total computation time is negligible. Therefore, CHS computation is approximately $N\times N$ times faster than the direct reference computation, where $N\times N$ is the segment size in pixels.

3. Acceleration of the ACPAS

A. Overview of the Graphics Processing Unit

The GPU technology allows us to use a set of highly parallel processors. A GPU is implemented as a set of multiprocessors. Each multiprocessor in the GPU has a single instruction multiple data (SIMD) architecture. At any given clock cycle, each processor of the multiprocessor executes the same instruction but operates on different data. The merits of implementation over the GPU platform are the high computational power and the low cost of the GPU board. NVIDIA (Santa Clara, California, USA) developed Compute Unified Device Architecture (CUDA) [16] so that their graphics products can be programmed directly into a high-level language. The CUDA is a new hardware and software architecture for issuing and managing computations on the GPU as a data-parallel computing device. In contrast with the multicore CPU architecture in which only a few threads

are executed concurrently, the NVIDIA CUDA technology can process more than thousands of threads simultaneously and then enables a higher capacity of information flow and massively parallel execution of instructions. For example, the GPU, NVIDIA Geforce GTX 285 graphic card used in this experiment has 30 streaming multiprocessors, and the maximum number of active threads per streaming multiprocessor is 768 [16]. From the software implementation point of view, the CUDA approach is similar to the traditional SIMD architecture.

On the other hand, although the new generation GPUs are flexible enough to be programmed for general-purpose computations, they were originally optimized for graphics applications. Therefore, to achieve high performance for holographic algorithms, thread level parallelism and memory access methods on multiprocessors in a GPU should be carefully considered. Another drawback of most GPUs is their lack of double-precision arithmetic support for floating point operations. Graphics processing does not typically require the same level of accuracy and precision as in holographic simulations. Hence, the loss of accuracy could be a concern in an application that runs on a GPU platform.

B. Computational Flow of the ACPAS on a Central Processing Unit

As mentioned in Section 2, the computation of the ACPAS is performed in two steps. In the first step, the contributions of object points to each segment are calculated. In more detail, spatial frequencies, phases, and phase compensations are determined, and the number of iterations during computation depends on the number of points and number of segments. In a second step, each segment is transformed by an IFFT. Therefore, the number of IFFT calculations depends on the number of segments. The pseudocode for the ACPAS algorithm, based on a CPU platform, is shown in Algorithm 1. Since the CPU is a serial processor, the total computational time is directly related to the total number of iterations.

Algorithm 1 ACPAS on CPU

```
Input data (points cloud) load
N_op ⇐ Number of the object points
N_seg ⇐ Number of the segments
for each object point of the points cloud do
for each segment of the fringe pattern do
Compute spatial frequency, amplitude phase, and phase compensation
end for
end for
for each segment of the fringe pattern do
Fringe ⇐ 2D IFFT (segment)
end for
```

C. Computational Flow of the ACPAS on a Graphics Processing Unit

In the first step of the ACPAS computational procedure, the contribution of each object point is limited to each segment, and, therefore, each segment can be processed independently. So the computation of spatial frequencies, amplitudes, phases, and phase compensations that correspond to each object point over each segment could easily be parallelized. Thus, computation of each task for the ACPAS on a GPU using a point cloud is parallelized. The second step is the IFFT for each segment. Since each segment is transformed independently, this step is also parallelizable. As described in Algorithm 2, the ACPAS computation can significantly benefit from a GPU based implementation.

Algorithm 2 ACPAS on GPU

```
Input data (points cloud) load
N_op ⇐ Number of the object points
N_seg ⇐ Number of the segments
for all i such that 0 ≤ i < N_op × N_seg in parallel do
Compute spatial frequency, amplitude, phase, and phase compensation
end for
for all i such that 0 ≤ i < N_seg in parallel do
Fringe ⇐ 2D IFFT (ith segment)
end for
```

4. Experiments and Results

A. Computational Speed

We have implemented the latest version ACPAS on a multi-GPU platform using CUDA and the computing environment and parameters are listed in Table 1. Comparative results for a 3 Mpixels holographic fringe pattern generation as a function of number of processed object points are shown in Fig. 1. Here the hologram size is $1\,\mathrm{K} \times 1\,\mathrm{K}$ pixels with three color channels for full color holography. The IFFT size is 64×64 pixels and the corresponding segment size is 32×32 pixels. Each measured computation time is the total for all computation steps including final normalization needed for the display. The solid curves show the computation time for the CPU implementation; the dotted curve represents the measured computation time for the GPU implementation. The computation time for the reference case is measured for one thousand object points; we can expect the computation time of the reference method to increase linearly with the number of object points [17]. In Fig. 1, the ACPAS on the CPU is four times slower than the CPAS on the CPU for one million object points. In addition, the ACPAS on the GPU is up to 300 times faster than the CPAS on the CPU and up to 130 thousand times faster than the reference hologram computation again for one million object points. Although the ACPAS on the GPU is slower than the CPAS on the GPU, the ACPAS on the GPU gives a higher quality reconstruction, which is similar to that of a reference hologram [15].

The computation of the ACPAS can be further accelerated by use of multiple GPUs. There are typically two methods as shown in Fig. 2. The first method, in Fig. 2(a), is based on partitioning the hologram plane according to the number of GPUs. Each partition is computed on an associated GPU. To form

Table 1. Computing the Environment and Parameters

Fringe pattern	Hologram size	1024 × 1024 pixels
	Segment size	32 × 32 pixels
	IFFT size	64 × 64 pixels
	Pixel interval	8 μm
	Distance between object center and hologram plane	500 mm
	Wavelength	633 nm
Computing system	CPU	Two Intel(R) Xeon(R) CPU 2 GHz
	Main memory	8 Gbytes
	GPU	Three NVIDIA Geforce GTX 280
Programming environment	Operating system	Linux 64 bit (Ubuntu 8.10)
	Programming language	Standard C and CUDA
	Libraries	CUFFTW

the final holographic fringe pattern, each generated fringe pattern is merged. The second method, in Fig. 2(b), is based on pipelining. Each frame of a fringe pattern is computed on the assigned GPU, and their computations are not synchronized with each other; this method gave the best performance. There are several reasons for this, with timing imbalance and competing access to computer resources as the main reasons. For the partitioning mode, the multiple GPUs being processed simultaneously can access the same resource at the same instant of time, resulting in unpredictable behavior. In addition, during merging of each generated fringe pattern, each GPU can be in the rest mode until the beginning point of the next task. Therefore, the pipelining mode is more efficient.

B. Accuracy of the ACPAS on the Graphics Processing Unit

All the CUDA based devices follow the IEEE-754 standard for binary floating-point arithmetic, including the Geforce GTX 280 used in this experiment. The single-precision and the double-precision representations allow 23 and 52 bits for the significand of a floating-point number, respectively. In many applications that use the GPU, a single-precision floating point is widely used to accelerate the computational performance, but this results in greater errors.

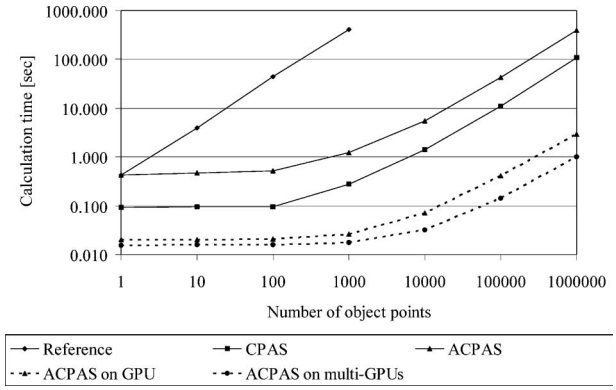

Fig. 1. Computational performance of the algorithms for a different number of object points.

The image difference between the ACPAS with double-precision floating-point arithmetic and the ACPAS with single-precision floating-point arithmetic for the impulse response is shown in Fig. 3. The normalized image difference in Fig. 3 is the absolute difference of the ACPAS using single precision and double precision, and the gray level of the normalized difference image indicates the phase difference between these two cases. Although these two fringe patterns have different phases, the two fringe patterns have quite similar local spatial frequency distribution. The difference in phase is caused by the limited number of bits allocated to the significand in the single-precision floating-point representation. Note that the phase computation is sensitive to errors. Therefore, the ACPAS on the GPU with single precision results in significant phase errors that are due to the limited number of bits.

To understand the effect of the phase error on the generated fringe pattern, the two kinds of ACPAS hologram generated by the single-precision arithmetic and double-precision arithmetic are compared in terms of the peak signal-to-noise ratio (PSNR), where the number of the points is the variable. In addition, the reconstruction from such holograms is also compared. The mean-square error (MSE) for two $m \times n$ holographic fringe patterns H_{sp} using single precision and H_{dp} using double precision, respectively, is determined as

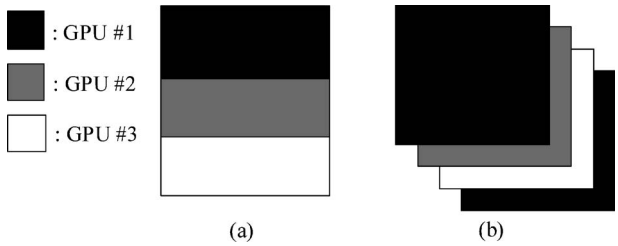

Fig. 2. Parallel digital hologram computation methods on a multi-GPU platform. The gray levels indicate the associated GPU number: (a) fringe pattern partitioned according to the available number of GPUs with each partition computed on the assigned GPU and (b) each fringe of a fringe sequence is computed on the assigned GPU.

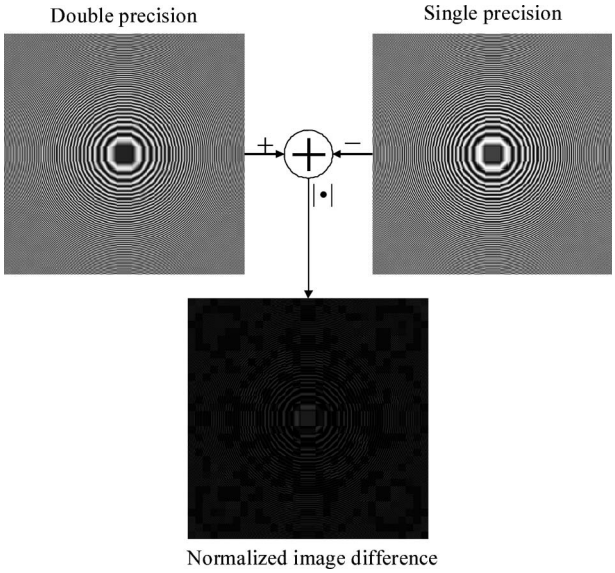

Fig. 3. Image difference between the ACPAS computed using double-precision and single-precision floating points.

$$\mathrm{MSE} = \frac{1}{mn}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}[H_{\mathrm{dp}}(i,j) - H_{\mathrm{sp}}(i,j)]^2, \qquad (5)$$

and the PSNR is computed as

$$\mathrm{PSNR} = 10\log_{10}\left(\frac{255^2}{\mathrm{MSE}}\right). \qquad (6)$$

Here $H_{\mathrm{dp}}(i,j)$ and $H_{\mathrm{sp}}(i,j)$ are normalized images whose pixel values are between $[0, 255]$, since most of the related display device operates at such quantization levels. Normalized pixel values of 0 and 255 correspond to lowest and highest values, respectively, among all the computed pixels of the low- and high-precision fringe patterns. Equations (5) and (6) are also used to measure the noise level of the reconstructions from the ACPAS using single-precision and double-precision arithmetic.

The experimental results are shown in Fig. 4. It can be seen that the PSNR between the high- and the low-precision fringe patterns is relatively constant at a level of approximately 25 dB. The PSNR difference between the reconstruction from high- and low-precision holograms decreases with the number of points. Point clouds, whose points are randomly distributed in three dimensions, are used as the input data, and each object point has a complex amplitude. The reconstruction from the two kinds of fringe pattern, ACPAS using single-precision arithmetic and ACPAS using double-precision arithmetic, is compared with the reconstruction from the reference hologram, and these results are shown in Fig. 5. As shown in Fig. 4, since the PSNR between the reconstruction from the ACPAS using single-precision and double-precision arithmetic decreases, one arithmetic may not be a suitable approach for hologram computation. However, as seen in Fig. 5, comparison with the reference case indicates almost the same result for both the single-precision and the double-precision ACPAS cases. In other words, the degradation in comparison with the reference case is almost the same for single- and double-precision ACPAS computations.

To understand this phenomenon, the reconstructions from the ACPAS on the CPU and the reference hologram are compared. The horizontal central profiles of the reconstructed images that correspond to a single object point are shown in Fig. 6, where the reconstructions from the ACPAS and the reference

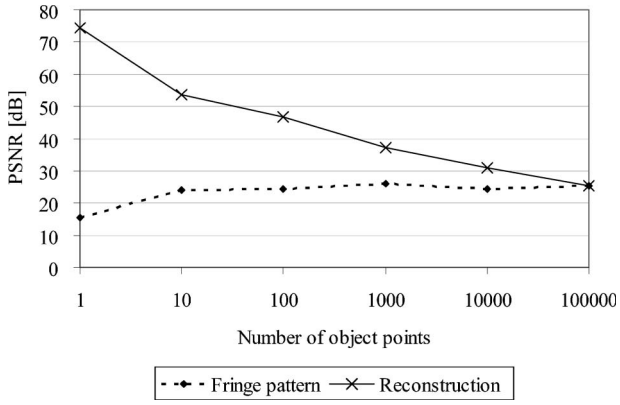

Fig. 4. Measured PSNR results corresponding to fringe patterns generated by using the ACPAS and the reconstruction from them. The dotted curve indicates the PSNR between low- and high-precision fringe patterns; the solid curve indicates the PSNR between the reconstructions from these low- and high-precision holograms. These reconstructions are normalized to be $[0, 255]$ by using the lowest and highest values among their fringe patterns. As the number of points increase, other distortion and noise factors dominate the computational noise that is due to lower precision.

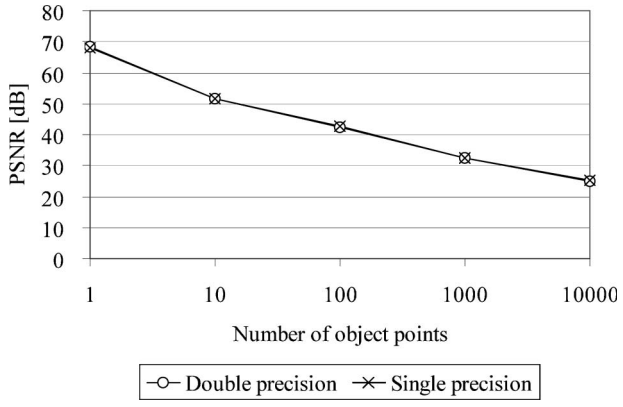

Fig. 5. Measured PSNR corresponding to different floating-point precision and the high-precision ACPAS hologram. The circles on the solid curve indicate the PSNR between the reconstructions from the reference hologram and the high-precision ACPAS hologram. The X on the solid curve represents the PSNR between the reconstructions from the reference hologram and the low-precision ACPAS hologram. In both cases these reconstructions are normalized by using the lowest and highest values among their fringe patterns. The error that is due to the ACPAS approximation is dominant; the additional error that is due to computational noise as a consequence of lower precision is negligible.

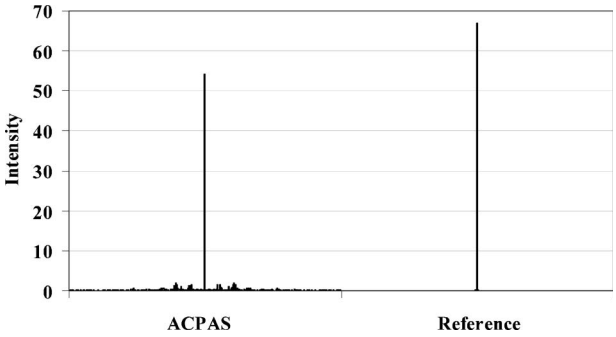

Fig. 6. Horizontal central profiles of the reconstructed images that correspond to a single object point.

hologram are shown at the left- and right-hand sides, respectively. The highest peak in each measured intensity distribution is the intensity of the reconstructed object that corresponds to the input data (a single object point), and the remainder of the intensity around the peak is the distortion, which is from the approximations used in the ACPAS algorithm. Even if the phase compensation method and a larger IFFT size reduces the inaccuracies, the distortion is not removed completely. If the GPU processor is used to compute the ACPAS, a computational noise is caused by a low number of bits for the significand. Moreover, as the number of points increase, this noise increases in magnitude. Therefore, as shown in Fig. 5, as the number of points increase, the PSNR decreases.

Our experiments show that the algorithmic distortion as outlined in the above paragraph is dominant; therefore, the additional distortion introduced by single-precision arithmetic over double-precision arithmetic is negligible. We also see that the reconstruction from single-precision holograms has a satisfactory quality. Thus, the benefit that is due to speed up on a GPU implementation (single precision) can be enjoyed without a disturbing loss in quality.

The numerically reconstructed images from the reference hologram, ACPAS on CPU and ACPAS on GPU, are shown in Fig. 7. The generated fringe patterns are amplitude-only holograms, and Fresnel diffraction is used as the numerical reconstruction algorithm. The point cloud, which consists of 1000 points uniformly distributed in a 10 mm × 10 mm × 10 mm cubic volume placed at a distance of 700 mm from the hologram, is used as the input data. By using the random point cloud, we can understand the reconstruction characteristics at any position in the space. There are both in-focus and out-of-focus object points in the reconstructions. As seen in Fig. 7, reconstruction from the ACPAS on the GPU, which performs single-precision arithmetic, is comparable with the ACPAS on a CPU and the reference case. Therefore, the ACPAS computation can significantly benefit from the speed associated with the GPU based implementations.

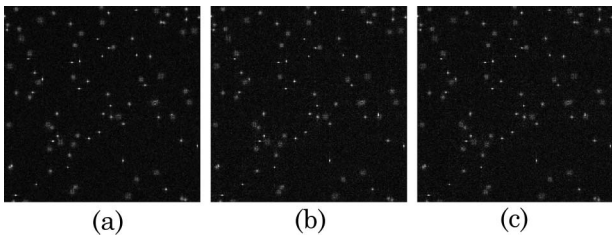

Fig. 7. Reconstruction of a random point cloud from (a) the reference hologram, (b) the double-precision ACPAS, and (c) the single-precision ACPAS.

5. Conclusion

Computation of the ACPAS on a GPU platform is approximately 100 times faster than a CPU based implementation. The computational accuracy on GPU implementations could be a concern. Even though the phase distribution on the fringe pattern using a single-precision floating point is quite different in comparison with the fringe pattern using a double-precision floating point, the visual reconstruction qualities are comparable. Therefore, single-precision implementations are adequate, and further acceleration can be achieved.

Moreover, a pipelined hologram computational method on a multi-GPU platform is discussed for further acceleration. With this method the computational speed of the ACPAS on the multi-GPU platform is up to 2.5 times faster in comparison with the single GPU platform, and a full color digital hologram can be generated at a rate of 30 frames/s for objects with more than 10,000 points. The ACPAS implementation on a multi-GPU platform is much faster without any significant degradation in reconstruction quality in comparison with the ACPAS running on the CPU. This computational performance might be adequate for real-time electroholographic display systems. The additional distortion that is due to computational noise associated with single-precision arithmetic is negligible in comparison with the algorithmic distortions inherent in the ACPAS method.

This research is supported by the European Commission within FP7 under grant 216105 with the acronym Real 3D.

References

1. Y. Ichihashi, H. Nakayama, T. Ito, N. Masuda, T. Shimobaba, A. Shiraki, and T. Sugie, "Horn-6 special-purpose clustered computing system for electroholography," Opt. Express **17**, 13895–13903 (2009).
2. J. A. Watlington, M. Lucente, C. J. Sparrell, V. M. Bove, and I. Tamitani, "A hardware architecture for rapid generation of electro-holographic fringe patterns," Proc. SPIE **2406**, 172–183 (1995).
3. T. Okada, S. Iwata, O. Nishikawa, K. Matsumoto, H. Yoshikawa, K. Sato, and T. Honda, "The fast computation of holograms for the interactive holographic 3d display system," Proc. SPIE **2577**, 33–40 (1995).
4. M. Lucente, "Interactive computation of holograms using a look-up table," J. Electron. Imaging **2**, 28–34 (1993).

5. M. Lucente, "Diffraction-specific fringe computation for electro-holography," Ph.D. dissertation (Massachusetts Institute of Technology, 1994).
6. H. Yoshikawa, "Fast computation of Fresnel holograms employing difference," Opt. Rev. **8**, 331–335 (2001).
7. M. Yamaguchi, H. Hoshino, T. Honda, and N. Ohyama, "Phase-added stereogram: calculation of hologram using computer graphic technique," Proc. SPIE **1914**, 25–33 (1993).
8. H. Yoshikawa and H. Kameyama, "Integral holography," Proc. SPIE **2406**, 226–234 (1995).
9. J. Tamai and H. Yoshikawa, "Faster computation of subsampled coherent stereogram," J. ITEJ **50**, 1612–1615 (1996). in Japanese, http://ci.nii.ac.jp/naid/110003336607.
10. H. Kang, T. Fujii, T. Yamaguchi, and H. Yoshikawa, "The compensated phase-added stereogram for real-time holographic display," Opt. Eng. **46**, 095802 (2007).
11. H. Kang, T. Yamaguchi, and H. Yoshikawa, "Accurate phase-added stereogram to improve the coherent stereogram," Appl. Opt. **47**, D44–D54 (2008).
12. H. Kang, F. Yaraş, L. Onural, and H. Yoshikawa, "Real-time fringe pattern generation with high quality," in *Digital Holography and Three-Dimensional Imaging* (Optical Society of America, 2009), paper DTuB7.
13. H. Kang, T. Yamaguchi, H. Yoshikawa, S. C. Kim, and E. S. Kim, "Acceleration method of computing a compensated phase-added stereogram on a graphic processing unit," Appl. Opt. **47**, 5784–5789 (2008).
14. L. Ahrenberg, A. J. Page, B. M. Hennelly, J. B. McDonald, and T. J. Naughton, "Using commodity graphics hardware for real-time digital hologram view-reconstruction," J. Disp. Technol. **5**, 111–119 (2009).
15. H. Kang, F. Yaraş, and L. Onural, "Quality comparison and acceleration for digital hologram generation method based on segmentation," in *3DTV CONFERENCE 2009* (IEEE, 2009).
16. "http://www.nvidia.com/,".
17. M. Lucente, "Optimization of hologram computation for real-time display," Proc. SPIE **1667**, 32–43 (1992).