Low memory distributed reconstruction of large digital holograms

Andrew J. Page[1], **Lukas Ahrenberg**[1] **and Thomas J. Naughton**[1,2]

[1] *Department of Computer Science, National University of Ireland, Maynooth, County Kildare, Ireland.*
[2] *University of Oulu, RFMedia Laboratory, Oulu Southern Institute, Vierimaantie 5, 84100 Ylivieska, Finland.*

Abstract: We present a parallel implementation of the Fresnel transform suitable for reconstructing large digital holograms. Our method has a small memory footprint and utilizes the spare resources of a distributed set of desktop PCs connected by a network. We show how we parallelize the Fresnel transform and discuss how it is constrained by computer and communication resources. Finally, we demonstrate how a 4.3 gigapixel digital hologram can be reconstructed and how the efficiency of the method changes for different memory and processor configurations.

OCIS codes: (090.1995) Digital holography; (100.2000) Digital image processing.

References and links

1. T. J. Naughton, Y. Frauel, B. Javidi, and E. Tajahuerce, "Compression of digital holograms for three-dimensional object reconstruction and recognition," Appl. Opt. **41**, 4124–4132 (2002).
2. B. Munjuluri, M. Huebschman, and H.R.Garner, "Rapid hologram updates for real-time volumetric information displays," Appl. Opt. **44**, 5076–5085 (2005).
3. L. Ahrenberg, P. Benzie, M. Magnor, and J. Watson, "Computer generated holography using parallel commodity graphics hardware," Opt. Express **14**, 7636–7641 (2006).
4. N. Masuda, T. Ito, T. Tanaka, A. Shiraki, and T. Sugie, "Computer generated holography using a graphics processing unit," Opt. Express **14**, 603–608 (2006).
5. M. Reicherter, S. Zwick, T. Haist, C.Kohler, H. Tiziani, and W. Osten, "Fast digital hologram generation and adaptive force measurement in liquid-crystal-display-based holographic tweezers," Appl. Opt. **45**, 888–896 (2006).
6. T. Ito, N. Masuda, K. Yoshimura, A. Shiraki, T. Shimobaba, and T. Sugie, "Special-purpose computer HORN-5 for a real-time electroholography," Opt. Express **13**, 1923–1932 (2005).
7. N. Masuda, T. Ito, K. Kayama, H. Kono, S. Satake, T. Kunugi, and K. Sato, "Special purpose computer for digital holographic particle tracking velocimetry," Opt. Express **14**, 587–592 (2006).
8. B. Hennelly and J. Sheridan, "Fast numerical algorithm for the linear canonical transform," J. Opt. Soc. Am. A **22**, 928–937 (2005).
9. T. Janse, V. von Rymon-Lipinski, N. Hanssen, and E. Keeve, "Fourier volume rendering on the GPU using a split-stream-FFT," in *Proc. of the VMV'04, Stanford, CA*, pp. 395–403 (IOS Press BV, 2004).
10. M. C. Pease, "An Adaptation of the Fast Fourier Transform for Parallel Processing," J. ACM **15**(2), 252–264 (1968).
11. Y. Frauel, T. J. Naughton, O. Matoba, E. Tajahuerce, and B. Javidi, "Three-dimensional imaging and processing using computational holographic imaging," Proc. IEEE **94**, 636–653 (2006).
12. T. Kreis, *Handbook of Holographic Interferometry* (Wiley-VCH, 2005).
13. D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods* (Athena Scientific, 1997).
14. T. M. Keane, A. J. Page, T. J. Naughton, S. A. A. Travers, and J. O. McInerney, "Building large phylogenetic trees on coarse-grained parallel machines," Algorithmica **45**, 285–300 (2006).

#89442 - $15.00 USD Received 7 Nov 2007; revised 12 Jan 2008; accepted 13 Jan 2008; published 29 Jan 2008

1. Introduction

In digital holography, optically captured digital holograms are usually reconstructed by algorithmic means using a computer. However, with continuing advances in CCD technology, novel methods are required to keep pace with the growing volume of data [1], and the associated increased computational requirements. In processor technology, the number of computations a single processor can perform per second has stagnated and recent developments have focused on adding multiple cores to a CPU. Thus, using a single powerful processor may no longer keep pace with the increased computational requirements for digital holography. In this paper we address this problem by implementing a parallel algorithm for digital hologram reconstruction suitable for distributed computing systems as well as multi-core processors.

While some effort has gone into accelerating computer generated holograms using algorithmic means [2], graphics processing units [3, 4, 5] and specialized hardware [6, 7], digital reconstruction of optically captured holograms has received little attention. A discussion on algorithmic optimization to Fresnel-like transforms has been presented [8], however due to the fact that the Fresnel transform is very efficiently implemented using the fast Fourier transform (FFT) there has been little need for special-purpose methods. It can be expected that the computational cost and memory requirements for hologram reconstruction will increase at a higher rate than computer technology improvements in the future. To address this we have constructed a Fresnel method that uses the spare processing and memory resources of a distributed set of desktop PCs to reconstruct large holograms.

To our knowledge this is the first time the Fresnel transform has been parallelized on a distributed system. While parallel versions of the FFT have been proposed [9, 10], none of them have been used to implement hologram reconstruction in a distributed system.

The paper is organized as follows. In Sect. 2 we present the algorithm. We analyze the limits of this algorithm for the parallelized Fresnel transformation in Sect. 3. In Sect. 4 we present experimental results showing a low memory reconstruction on a single processor and speedup achieved using multiple processors, and we conclude in Sect. 5 by discussing our results and some ideas for future research.

2. Fresnel transform

A reconstruction R from a digital hologram H at distance d can be computed using the following discrete Fresnel transform [11, 12] to simulate the propagation of light

$$R(m'\Delta x', n'\Delta y'; d) = \mathscr{F}^{-1}\Bigg(\mathscr{F}\big[H(m\Delta x, n\Delta y)\big] \times \exp\left\{-\mathrm{i}\pi\lambda d\left[\frac{u^2}{(\Delta x N_{\mathrm{x}})^2} + \frac{v^2}{(\Delta y N_{\mathrm{y}})^2}\right]\right\}\Bigg), \tag{1}$$

where (m,n) and (m',n') are discrete spatial coordinates in the hologram plane and reconstruction plane, respectively, $(\Delta x, \Delta y)$ and $(\Delta x', \Delta y')$ are the spatial resolutions in the hologram and reconstruction planes, respectively, N_{x} and N_{y} are the number of samples in the x and y directions, respectively, λ is the wavelength of the light, $\mathscr{F}$ denotes a two-dimensional Fourier transform, and u and v are discrete spatial frequencies. This particular form is useful for near-field analysis when one requires the same resolution at each reconstruction distance.

2.1. Parallelized Fresnel transform

The 2D Fourier transform is linearly separable into two orthogonal 1D Fourier transforms. As depicted in Fig. 1, the algorithm consists of three stages. Each stage must be fully completed before the next stage can proceed. In stage 1, a 1D FFT is performed on each row of the

#89442 - $15.00 USD Received 7 Nov 2007; revised 12 Jan 2008; accepted 13 Jan 2008; published 29 Jan 2008

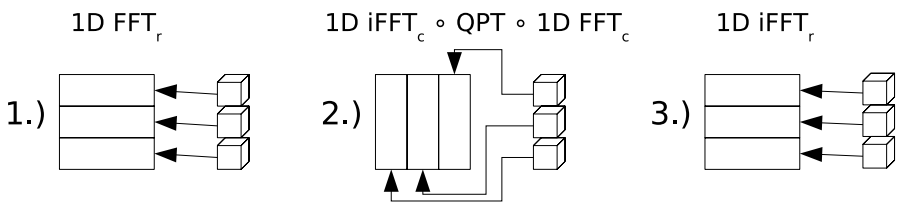

Fig. 1. Three stage parallelized reconstruction algorithm based on Eq. (1). The text shows the computations performed, where QPT denotes multiplication by the quadratic phase term of Eq. (1). The cubes represent processors operating on rows or columns individually.

hologram. In stage 2, a 1D FFT is performed on each column, the quadratic phase factor of Eq.(1) is applied, and a 1D inverse FFT is performed on each column. Finally, in stage 3, an inverse 1D FFT on each row reveals the reconstructed hologram.

3. Limits on holographic parallelization

In this section, we discuss how our parallel algorithm is affected by the available computer resources. An introduction and general discussion on parallel computing can be found in Ref. [13]. In practical terms, parallelizing Eq. (1) is constrained by a number of factors which limit the efficiency of the parallelization and achievable speedup. These limits are: 1) available memory, 2) available processing resources, and 3) a finite communications channel. Speedup refers to how much a parallel algorithm is faster than a corresponding sequential algorithm: calculated by dividing the sequential execution time by the parallel execution time. Efficiency is defined as the percentage utilization of the processors during the execution of the algorithm.

A multi-core CPU contains a number of processing units which are connected by a high bandwidth bus to shared memory. A loosely-coupled distributed system contains a number of independent processors, with no shared memory, and no shared clock, connected by a communications channel such as the Internet. We will only consider sets of homogeneous processors.

The limits on the applicability of parallelization for this algorithm are: the size of the reconstruction, the memory requirement of the reconstruction, the number of processors, the granularity of parallelization, and the rate of transmission of data. We will define these limits, to allow for a more efficient implementation and execution of the algorithm.

3.1. Granularity of parallelization

The coarsest granularity of parallelization is achieved by using a hologram row or column as an atomic unit and grouping these together to achieve the desired granularity. In the following section we assume a reconstruction with dimensions $N \times N$ pixels, and we have a parallel computing system consisting of P processors. The maximum degree of parallelization is N, thus the maximum possible speedup of the parallelized reconstruction will be achieved by using $P = N$ processors. Ignoring communication time (assuming instantaneous communication), there would be no idle time, resulting in 100% processor efficiency.

From Fig. 1, the amount of data to be transmitted is $6N^2$ pixels where there is N^2 pixels transmitted in each direction at each of the three stages of the computation. When $P < N$ the optimal number of rows to group together is $\lceil N/P \rceil$. This, however, may not be realistically achievable due to memory limitations, necessitating smaller row groups.

Breaking down the algorithm further where each hologram row is parallelized requires breaking up the FFT computation. This results in increased communication costs of $6N^2 \log_2 N$ pixels where there is an additional $\log_2 N$ transmission overhead from additional intermediate calcu-

#89442 - $15.00 USD Received 7 Nov 2007; revised 12 Jan 2008; accepted 13 Jan 2008; published 29 Jan 2008

lations. Most parts of the parallelized FFT are dependent on previously calculated data and the maximum degree of parallelization is $N^2/2$. If there is instantaneous communication, only N processors would be processing 100% of the time, with all other processors lying idle for significant periods of time. The efficiency at the maximum degree of parallelisation ($P = N^2/2$), is calculated as

$$\log_2 N/(N-1)\,. \tag{2}$$

There is no reduction in execution time when $P > N^2/2$. If $N = 32$ and $P = 512$, then the processor efficiency is 16% using Eq.(2).

These upper bounds on speedup are not realistically achievable because communication costs are never instantaneous and it is rarely feasible to use large amounts of computing resources inefficiently. Thus we recommend that, firstly, whole rows and whole columns are used as the smallest granularity of parallelization, to minimize communication costs and increase processor efficiency, and secondly, no more than $P = N$ processors be employed (achieving simultaneously high speedup and high efficiency).

3.2. Reconstruction size

For a given set of processing and communication resources, we find the bound on hologram size for which a reconstruction takes the same length of time on a parallel system as it does on a single machine. We assume a dedicated client-server architecture with a single shared communications channel, with 100% processor efficiency and the granularity of parallelization is at the row level.

A holographic pixel can be represented in many different formats, so we abstract away from implementation specific details. The number of holographic pixels to transmit is $6N^2$. The speed of the communications channel, B, is defined in terms of the number of hologram pixels transmitted per second, which abstracts away from implementation specific issues, such as compression. Thus, the total transmission time is $6N^2/B$.

The algorithm requires $mN^2 + fN^2\log_2 N + N$ calculations, where m is the number of calculations to generate the quadratic phase term of Eq. (1) and f is a constant factor based on the implementation of the FFT. The point at which a parallel system takes the same length of time as a single machine is thus when $mN^2 + fN^2\log_2 N + N = \{(mN^2 + fN^2\log_2 N + N)/P\} + 6N^2/B$ is satisfied, where P is the number of processors used. Ignoring constants, this relationship has the simplified interpretation of

$$N = 2^{1/P+1/B}, \tag{3}$$

which describes the dependency of the minimum reconstruction size required to benefit from parallelization. As B or P increases, N asymptotically decreases towards a lower bound. This equation can be reworked to calculate other parameters such as the minimum transmission rate and the minimum number of processors. Speedup, S, can be calculated by setting $S = (mN^2 + fN^2\log_2 N + N)/[\{(mN^2 + fN^2\log_2 N + N)/P\} + 6N^2/B]$, which is an upper bound on the speedup possible for a given setup and problem instance. As the size of the reconstruction required increases, so does the efficiency of the resource utilization.

4. Experimental results

We have implemented the reconstruction algorithm on a Java based distributed system, which uses the spare clock cycles of idle PCs in a university teaching laboratory [14]. Based on this implementation we have evaluated the parallelism of our method as well as the efficiency of memory constraints as described in Sect. 4.1 and Sect. 4.2 below.

#89442 - $15.00 USD Received 7 Nov 2007; revised 12 Jan 2008; accepted 13 Jan 2008; published 29 Jan 2008

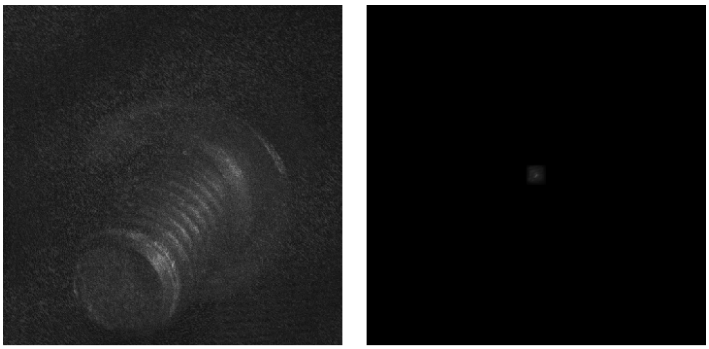

Fig. 2. Animation of the $2^{16} \times 2^{16}$ reconstruction, from (a) zoomed-in view to (b) full field (330 KB - QuickTime).

In order to test the performance for large holograms we reconstructed a $2^{16} \times 2^{16}$ (4.3 gigapixel) digital hologram. This is, to our knowledge, the largest ever digital hologram reconstructed. Since no real digital holograms of this size have been captured, we padded out a 2032×2048 hologram in the hologram plane. We used a computer with a single 2.2 GHz Xeon processor and 1 GB of memory, running GNU/Linux. The total reconstruction time on our system was 30 hours. In Fig. 2(a) a zoomed in view of the centre of the reconstruction plane can be seen. Figure 2(b) shows the relative size of the object within the full field.

4.1. Distributed reconstruction time

We have used multiple processors to decrease the total reconstruction time of a large digital hologram reconstruction. Figure 3(a) shows the reduction in reconstruction time achieved when using multiple processors. We used a homogeneous set of 26 desktop PCs running GNU/Linux, each with 2.0 GHz Intel Processors, 1 GB of memory and connected by a non-dedicated 100 Mb/s Ethernet network. The total processing time is reduced by utilizing more processors, however only to a certain point, which varies depending on the computation rate of the processors, the speedup of the network and the size of the reconstruction. In Fig. 3(a) there are large reductions in the reconstruction time when up to 8 processors are added. After that there is no benefit gained by using more processors (as explained in Sect. 3), because the network connection is being fully utilized by the system.

4.2. Low memory reconstruction

Reconstruction computations have a large space requirement. As the size of a reconstruction increases, it quickly becomes infeasible to process the whole reconstruction in memory on a standard PC. We utilize the high capacity of low cost commodity hard disks in lieu of increased memory. We only keep the portion of the reconstruction which is currently being processed in memory, with the rest of the data stored to hard disk. There is an additional overhead to read and write data on a hard disk, but since we have advanced knowledge of how the data needs to be accessed, we can minimize this factor. The overhead varies depending on the hardware and the amount of data stored in memory at any one time.

If more memory is available, it is more efficient to read in and process multiple rows at once, with this grouping of rows referred to as a unit of work. Figure 3(b) shows the variation in processing times with different unit sizes when reconstructing a $2^{12} \times 2^{12}$ hologram, using a single 2.2 GHz Xeon processor with 2 GB of memory. We performed the reconstructions with

#89442 - $15.00 USD Received 7 Nov 2007; revised 12 Jan 2008; accepted 13 Jan 2008; published 29 Jan 2008

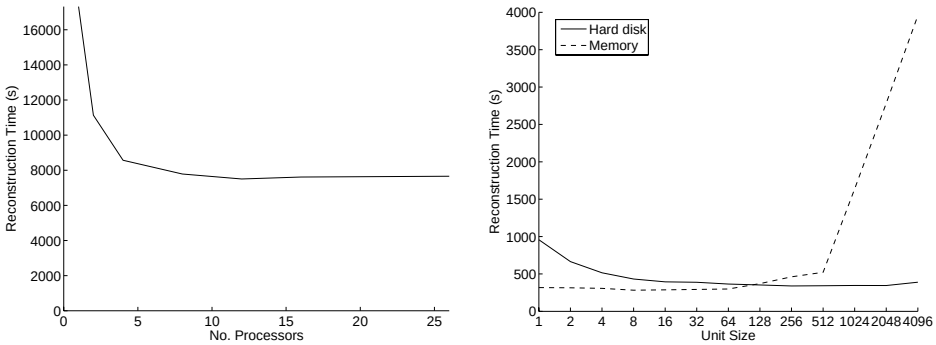

Fig. 3. Results: (a) Reconstruction time for $2^{14} \times 2^{14}$ digital hologram using varying numbers of processors. (b) Execution time with varying sized units for a $2^{12} \times 2^{12}$ reconstruction, using the hard disk as intermediate storage versus keeping the whole computation in memory.

and without (i.e. using memory only) using the hard disk as intermediate storage. Using the hard disk, we found that increasing the unit size reduces the reconstruction time by more efficiently reading and writing to the hard disk. Keeping the whole computation in memory provides a consistant reconstruction time, which is faster than using the hard disk as intermediate storage, but only to a point. When the available physical memory is expended, non-optimized hard disk based swap space is used by the operating system, increasing the reconstruction time eight-fold.

With a unit size of one row, the hard disk based reconstruction algorithm requires 16 MB of memory compared to 800 MB when storing the whole hologram in memory. This memory limitation prevents the memory-only based algorithm from working at all for large holograms. However, with the hard disk based reconstruction algorithm, simply choosing a unit size which falls within the available memory of the machine makes it possible to reconstruct very large holograms on standard commodity hardware.

5. Conclusions

We have created a parallel Fresnel hologram reconstruction method that can reconstruct large holograms on standard desktop PCs. We have shown how it is possible to reduce the reconstruction time for large holograms by using a distributed system. The method also has a small memory footprint, allowing for the possibility of performing holographic reconstructions on resource constrained devices. This could open up possibilities for shared distributed computing on, for example, mobile devices in the future.

In our future work we will look at the effect of using a heterogeneous web computing system for reconstructions, as well as robust parallel reconstructions with quality of service guarantees for holographic video. Other interesting possibilities include implementing other computationally expensive methods in digital holography, for example advanced speckle reduction and hologram image processing techniques.

Acknowledgments

We thank Conor Mc Elhinney for capturing the digital holograms. We acknowledge support from Science Foundation Ireland, Academy of Finland, and the European Commission through a Marie Curie Fellowship.

#89442 - $15.00 USD Received 7 Nov 2007; revised 12 Jan 2008; accepted 13 Jan 2008; published 29 Jan 2008