

文章编号:1007-2780(2019)10-1021-09

GPU 加速电子剪切散斑干涉图像处理

邵 珩*, 周 勇, 祁俊峰, 聂中原

(北京卫星制造厂有限公司, 北京 100094)

摘要:电子剪切散斑干涉是在电子散斑干涉基础上发展起来的一种测量位移导数的新方法,尤其适合于无损检测实时测量。由于噪声干扰,测量得到的相位差图有大量噪点需要滤波去除。滤波处理计算量大,采用 CPU 计算需要很长时间完成。为缩短滤波时间,实现实时滤波处理,对四步相移法和 Butterworth 低通滤波、正余弦均值滤波和复数均值滤波算法,利用 GPU 并行计算能力强和存储带宽高的特性,开发了基于 GPU 加速计算的解相位差和滤波算法。对比基于 CPU 的解相位差和滤波计算, GPU 加速计算将图像总体处理时间约 1.9 s (i7 6500U) 降低到了 239~366 ms (GeForce 940MX) 或 86~116 ms (Quadro P3000),且正余弦均值滤波和复数均值滤波算法均可以通过降低滤波次数缩短滤波时间,从而保证在性能较弱的 GPU 上也能将计算时间压缩到 0.2 s 内,满足实时图像处理的需求。

关键词:电子剪切散斑干涉,图像处理,滤波, GPU 加速计算

中图分类号: TP312 **文献标识码:** A **doi:** 10.3788/YJYXS20193410.1021

GPU accelerated image processing for electronic speckle pattern shearing interferometry

SHAO Heng*, ZHOU Yong, QI Jun-feng, NIE Zhong-yuan

(Beijing Satellite Manufacturing Co., Ltd BSC, Beijing 100094, China)

Abstract: Electronic speckle pattern shearing interferometry (ESPSI) is a new method for measuring displacement derivatives based on electronic speckle pattern interferometry. It is especially suitable for real-time measurement of nondestructive testing. The acquired phase difference map in the ESPSI is usually seriously affected by noise. Thus filtering processing is needed to get a good phase difference map. Filtering by CPU takes too long time for real-time processing. To take full advantage of GPU's strong parallel computing capability and high storage bandwidth, GPU-accelerated phase difference calculation and filtering algorithm based on four-step phase-shift method and Butterworth low-pass filter, sine-cosine mean filter and complex mean filter methods were developed to accelerate image processing. GPU-accelerate processing shortened the processing time from about 1.9 s (i7 6500U) to 239~366 ms (GeForce 940MX) or 86~116 ms (Quadro P3000). The time cost by sine-cosine mean filter and complex mean filter methods can be further reduced to less than 0.2 s by filtering fewer times to meet the needs of real-time ESPSI meas-

收稿日期: 2019-04-22; **修订日期:** 2019-06-10.

基金项目: 国家重点研发计划重大科学仪器设备开发重点专项 (No.2016YFF0101800)

Supported by Key Projects for the Development of Major Scientific Instruments and Equipment of National Key R&D Program (No.2016YFF0101800)

* 通信联系人, E-mail: shao_heng@126.com

urement when the performance of the GPU is weak.

Key words: electronic speckle pattern shearing interferometry; image processing; filtering; GPU accelerated calculation

1 引言

电子剪切散斑干涉测量技术是一种全场、非接触、高精度和高灵敏度的光学测量方法,测量系统光路简单,调节方便,对环境要求不高,特别适合于无损检测实时测量^[1-3]。散斑图像的处理需要大量的运算。例如采用时间相移法时,首先需要将若干散斑图进行相减、相除和反正切计算,得到散斑干涉相位差图,然后通过滤波处理去掉噪声,获得散斑条纹图,最后通过解包裹算法得到物体变形量的空间导数。随着激光散斑干涉测量技术的应用,相关滤波算法也得到较多研究^[4-8],但这些研究一般都关注于改善滤波效果,对滤波时间基本不做考虑,无法应用于电子剪切散斑干涉连续测量的实时处理。在电子剪切散斑干涉连续测量中,需要计算机在短时间内对硬件获得的大量图像数据进行处理,因此计算机的运算能力成为能否实时处理测量数据的关键。传统的 CPU 计算灵活性高,适合处理各种复杂情况,但 CPU 单核运算能力有限,随着图像处理规模的不断扩大,CPU 的处理能力越来越成为测量结果实时处理显示的瓶颈;若采用多核 CPU 计算,一则计算任务的分割、多核 CPU 间的通信处理较为繁琐,二则随着核心数量的增多,CPU 平台的成本也迅速提高,不利于无损检测实时测量的商业应用。

GPU 传统上用于游戏图形显示,近年来因其运算能力、存储带宽远超 CPU 而被用于加速计算。GPU 计算核心数量众多而单核处理能力较弱,尤其适用于大规模并行计算任务,被广泛应用于流体动力学模拟^[9-10],分子动力学模拟^[11-12],神经网络与深度学习算法等^[13-15]。由于 GPU 具有强大的计算能力,易于实现数据的实时处理^[16-17],因此,将 GPU 用于图像处理有望大幅提升计算机的运算能力,实现电子剪切散斑干涉的实时测量;同时,GPU 的运算能力和能源效率远超同价位的 CPU,有利于相关商业平台的开发。本研究拟在电子剪切散斑干涉连续测量的数据处

理中采用 GPU 加速计算技术,以提高图像处理效率,提高测量效率。

2 电子剪切散斑干涉原理

激光照射到粗糙物体表面时发生漫反射,生成无数子波。这些子波在空间中发生干涉,形成明暗不一的斑点,称为激光散斑。由于激光散斑源于激光在粗糙表面上发生的漫反射,携带了粗糙表面的形貌信息,当粗糙表面的形貌发生变化时,激光散斑的分布也随之发生变化,因而可以利用激光散斑的这一特性对物体表面的形变进行测量。

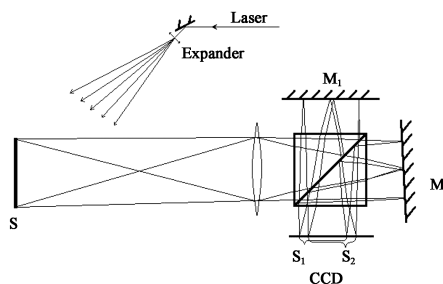


图 1 基于迈克尔逊剪切装置的剪切散斑干涉光路图
Fig. 1 Shearing speckle interferometry optical system based on Michelson shearing device

图 1 为剪切干涉光路示意图。相干光经粗糙被测物 S 漫反射,部分光线被分光镜反射后经反射镜 M_1 再次反射,穿过分光镜后进入 CCD 成实像 S_1 ;另一部分光线穿过分光镜,被反射镜 M_2 反射后经分光镜反射进入 CCD 成实像 S_2 。如果 M_1 和 M_2 相互垂直并且到分光镜的距离相等时, S_1 与 S_2 大小相等且互相重合;当反射镜 M_2 的角度发生偏转时, S_2 的位置将发生移动,与 S_1 形成剪切。将 S_1 与 S_2 之间距离的大小和方向映射到被测物上即得到剪切量和剪切方向。此时, S_1 和 S_2 将在像面上互相干涉,生成散斑图像。若记录被测物发生形变前后的散斑干涉图像,经过处理可以得到表示物体沿剪切方向的位移偏导数的条纹图案。

3 GPU 图像处理方法

GPU 加速计算将计算任务分派给大量的线程同时处理,利用远高于 CPU 的线程数量获得远超出 CPU 的性能。GPU 本身单核计算能力很弱,因此只有将计算任务分割为互相独立的小任务,使每个线程都可以不依赖其他线程的计算结果而完成本身的计算任务时,才能发挥 GPU 线程数量多的优势。

采用 CUDA 平台处理计算任务时,CPU 作为控制器,负责读取数据,并将数据传输给 GPU;GPU 作为协处理器,接收 CPU 传递的数据,在显存中开辟相应存储空间,存储传入的数据和中间处理数据,将这些数据分配给 GPU 上的多个线程进行处理;处理完成后,将数据传回 CPU,由 CPU 进行后续处理。

电子剪切散斑干涉实时测量过程中,图像处理包括两部分:解相位差和滤波。解相位差采用四步相移法^[18]:首先在加载前对被测物采集 4 幅散斑图,每次采集后对实像 S_1 引入 $\pi/2$ 的相移。4 幅图的光强可以表示为:

$$I_i = a + b \cos\left(\alpha + \frac{i-1}{2}\pi\right), \quad (1)$$

式中: I_i 为第 i 步测得的光强分布, $i=1,2,3,4$, a 为背景光强, b 为调制度, α 为未知的随机相位。

加载后,再次采集 4 幅散斑图,光强为:

$$I_i' = a + b \cos\left(\alpha + \beta + \frac{i-1}{2}\pi\right), \quad (2)$$

式中: β 为加载引入的相位差。

根据 I_i 和 I_i' , 可以解出每个点在加载前后位置的相位差:

$$\theta_1 = \tan^{-1} \frac{I_4 - I_2}{I_1 - I_3}, \quad (3)$$

$$\theta_2 = \tan^{-1} \frac{I_4' - I_2'}{I_1' - I_3'}, \quad (4)$$

$$\theta = \theta_2 - \theta_1. \quad (5)$$

为滤除相位差图的噪声,采用 3 种速度较快的滤波方法,包括正余弦均值滤波、复数均值滤波、Butterworth 低通滤波方法,分别比较它们的精度、计算效率以及 GPU 加速效果。

正余弦均值滤波中,对相位差图任意点 (x, y) , 读取该点及邻域点的相位值 θ , 首先将其转换为相位的正余弦值 $\sin\theta$ 和 $\cos\theta$, 对正余弦值进行均值滤波后,通过 $\overline{\sin\theta}$ 和 $\overline{\cos\theta}$ 计算 $\bar{\theta}$ 。当采用 3×3 的滤波窗口时,具体算法如下:

$$\overline{\sin\theta}_{x,y} = \frac{1}{25} \left[5 \sin\theta_{x,y} + 3 (\sin\theta_{x-1,y} + \sin\theta_{x+1,y} + \sin\theta_{x,y-1} + \sin\theta_{x,y+1}) + 2 (\sin\theta_{x-1,y-1} + \sin\theta_{x-1,y+1} + \sin\theta_{x+1,y-1} + \sin\theta_{x+1,y+1}) \right], \quad (6)$$

$$\overline{\cos\theta}_{x,y} = \frac{1}{25} \left[5 \cos\theta_{x,y} + 3 (\cos\theta_{x-1,y} + \cos\theta_{x+1,y} + \cos\theta_{x,y-1} + \cos\theta_{x,y+1}) + 2 (\cos\theta_{x-1,y-1} + \cos\theta_{x-1,y+1} + \cos\theta_{x+1,y-1} + \cos\theta_{x+1,y+1}) \right]. \quad (7)$$

根据式(6)和式(7)得到 $\overline{\sin\theta}$ 和 $\overline{\cos\theta}$ 后,即可通过反正切函数将 $\bar{\theta}$ 还原到 $[0, 2\pi)$ 范围内。通常,一次滤波的效果并不能令人满意,需要经过多次滤波才能得到较为平滑的结果。若采用更大的滤波窗格,则在点 (x, y) 附近更大的邻域内计算得到相应的 $\overline{\sin\theta}$ 和 $\overline{\cos\theta}$ 。

对正余弦均值滤波方法采用 GPU 加速时,首先将图像分割为 32×32 的小块,将每个小块及其外围一圈像素分配给一个线程块,即采用 3×3

窗口时,每个线程块分配 34×34 的小块,其中超出图像边界的部分采用镜像处理。对小块内像素的相位值分别求正弦和余弦值,然后采用式(6)、(7)求得正余弦均值并通过反正切函数还原为相位。为提高计算效率,每个线程块内采用共享存储存取正余弦值。

复数均值滤波中,首先将相位值转换为复数:

$$c_{x,y} = \exp(\theta_{x,y}i), \quad (8)$$

然后按照类似式(6)的方式对其求取均值:

$$\overline{c_{x,y}^{n+1}} = \frac{1}{25} \left[5 \overline{c_{x,y}^n} + 3 (\overline{c_{x-1,y}^n} + \overline{c_{x+1,y}^n} + \overline{c_{x,y-1}^n} + \overline{c_{x,y+1}^n}) + 2 (\overline{c_{x-1,y-1}^n} + \overline{c_{x-1,y+1}^n} + \overline{c_{x+1,y-1}^n} + \overline{c_{x+1,y+1}^n}) \right], \quad (9)$$

式中 $\overline{c_{x,y}^n}$ 为点 (x,y) 处滤波 n 次得到的平均值, $\overline{c_{x,y}^0} = c_{x,y}$ 。

多次滤波完成之后再通过反正切函数将复数平均值还原为相位值。

比较正余弦均值滤波和复数均值滤波可以发现,在进行一次滤波时,两者是等价的。在进行多次滤波时,复数均值滤波减少了正余弦计算和反正切计算次数,降低了计算量。

对复数均值滤波进行 GPU 加速时,将相位与复数的转换与还原、复数求均值分别进行 GPU 加速计算,其中求均值部分采用类似正余弦滤波图像分割的方法采用共享存储加速计算。

Butterworth 低通滤波中,对相位差图任意点 (x,y) 的相位值 θ ,将其转换为复数 c 。对 $c_{x,y}$ 进行傅里叶变换,得到 $C_{u,v}$,然后进行低通滤波:

$$C_{u,v}^* = C_{u,v} / [1 + (D_{u,v}/D_0)^{2n}] \quad (10)$$

式中, $D_{u,v}$ 是到原点的距离, D_0 为截止距离, n 为滤波器的阶数。

对 $C_{u,v}^*$ 进行傅里叶反变换,得到 $c_{x,y}^*$,再通过反正切函数即可还原为相位值 θ 。

对 Butterworth 低通滤波采用 GPU 加速时,可以将转换与还原、正反傅里叶变换和低通滤波每个步骤均采用相应的 GPU 计算。

4 实验结果

4.1 相位差图生成

4.2.1 CPU 解相位差

采用 CPU 解相位差时,首先读取 8 幅图片的数据,然后扫描图片,解出每个点的相位差值。解相位差的全部时长约 500 ms,其中 25 ms 用于读取图片,465 ms 用于解相位差计算,10 ms 用于保存图片,解相位差的时间占总时间的 90% 以上。采用的 CPU 为 i7-6500U@2.5 GHz,图片尺寸为 $2\,592 \times 2\,048$,下同。图 2 为计算得到的相位差图。

4.2.2 GPU 解相位差

从式(3)~(5)可知,每个点的相位差求解过程都与周围点无关,因此适用于 GPU 并行处理。本研究中采用两个不同的 GPU 进行图像处理,其中 GPU1 型号为 GeForce 940MX @ 1004

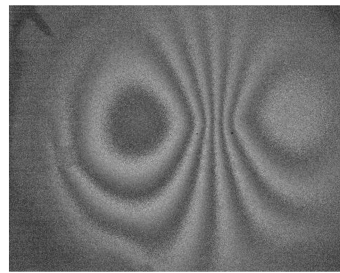


图 2 四步相移法得到的相位差图

Fig. 2 Phase-difference map obtained by four-step phase shift method

MHz, GPU2 型号为 Quadro P3000@1088 MHz, 下同。

求解结果表明,采用 GPU 解相位差的耗时远少于 CPU 解相位差,如表 1 所示。采用 GPU1 时,全部求解时长约 77 ms,其中 26 ms 用于读取图片,43 ms 用于解相位差计算,8 ms 用于保存图片,解相位差的时间下降到总时间的约 56%;采用 GPU2 时,解相位差时间下降到 18 ms,总耗时 52 ms。

表 1 GPU 加速解相位差与 CPU 解相位差耗时对比

Tab.1 Time consumption of phase difference calculation by CPU and GPU

ms					
耗时	读取	初始化	数据传输	求解	总计
CPU	25	—	—	465	500
GPU1	26	10	27	6	77
GPU2	26	7	9	2	52

进一步分析表明,对于 GPU1,在 43 ms 的解相位差时间中,GPU 初始化和分配显存耗时 10 ms,从 CPU 到 GPU 的数据传输耗时约 23 ms,计算耗时约 6 ms,数据从 GPU 传回至 CPU 耗时约 4 ms;对于 GPU2,初始化和分配显存耗时约 7 ms,从 CPU 到 GPU 的数据传输耗时约 7 ms,计算和数据回传耗时均约 2 ms。若仅考虑计算时间,GPU1 的计算速度是 CPU 单核的约 80 倍, GPU2 的速度则达到 200 倍以上。由此可见,在解相位差计算中,GPU 初始化和显存分配、数据传输占据了绝大部分时间,真正用于计算的时间非常短。

计算时间对比表明, CPU 解相位差的瓶颈在于求解过程,而采用 GPU 加速计算时,求解过

程耗时占比极小,绝大部分时间用于数据存取与传输,其次为 GPU 的初始化过程,计算总时间降到原来的约 10.4%~15.4%,大幅加速了求相位差图过程。

4.2 相位差图滤波处理

4.2.1 CPU 串行算法滤波

为便于比较不同滤波算法的效果,首先采用模拟散斑图进行测试。模拟相位分布为:

$$d_1 = \text{sqrt}[(x - 1696)^2 + (y - 1024)^2], \quad (11)$$

$$d_2 = \text{sqrt}[(x - 896)^2 + (y - 1024)^2], \quad (12)$$

$$\theta_{x,y} = 3000 \times [\exp(d_1/1000) - \exp(d_2/1000)]. \quad (13)$$

然后给模拟相位附加均值为 0、标准差为 60 的高斯噪声,得到含噪声散斑图。无噪声相位图和含噪声散斑图如图 3 所示。

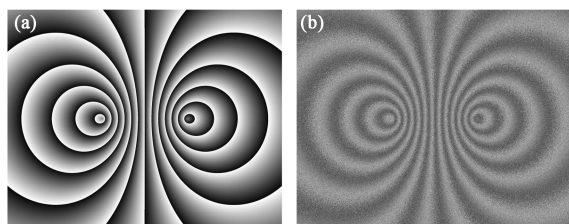


图 3 模拟相位图。(a)无噪声;(b)含噪声。

Fig.3 Simulated phase diagram. (a) Without noise; (b) With noise.

对图 3(b)采用不同的算法进行滤波,并比较其标准峰值信噪比 PSNR 的变化。其中 PSNR 的定义为:

$$K_{\text{PSNR}} = 10 \times \log\left(\frac{255^2}{K_{\text{MSE}}}\right), \quad (14)$$

式中: K_{MSE} 为滤波后图像与不含噪声的原始相位图之间的均方误差。

由定义可知,PSNR 越大,滤波后图像与原始相位图差别越小,滤波失真越小。图 4 为正余弦均值滤波和复数均值滤波的 PSNR 随滤波次数和滤波窗口尺寸变化的曲线及不同阶次 Butterworth 低通滤波的 PSNR 随截止距离的变化。

图 4 表明,对于正余弦均值滤波和复数均值滤波而言,滤波窗口越大,PSNR 上升越快,且在相同的滤波窗口下正余弦滤波效果较复数滤波更好。对于 Butterworth 低通滤波,阶次越低,达到

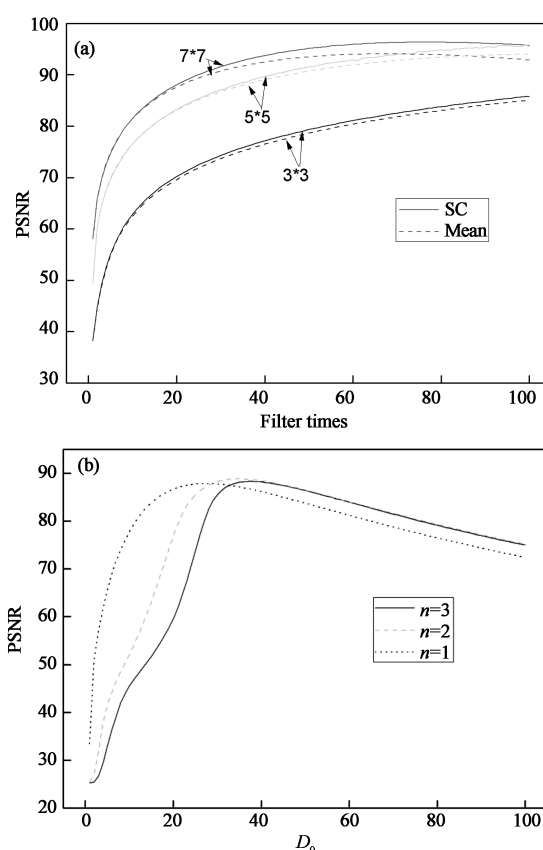


图 4 正余弦均值滤波(SC)和复数均值滤波(Mean)的 PSNR 随滤波次数和滤波窗口大小变化(a),与不同阶次 Butterworth 低通滤波的 PSNR 随截止距离的变化(b)。

Fig.4 Influence of the filtering times and window size on PSNR of sin-cos filtering (SC) and complex mean filtering (Mean) (a), and influence of the order and cut-off distance on PSNR of Butterworth filtering (b).

最佳效果的截止距离越短。总体而言,正余弦均值滤波和复数均值滤波的最好效果好于 Butterworth 低通滤波。由于 Butterworth 滤波的计算时间比较稳定,不随阶次和截止距离而变化,以其达到的最佳效果为标准($n = 2$ 、 $D_0 = 37$ 时, PSNR 为 88.84),选取 $\text{PSNR} \geq 88.84$ 的正余弦均值滤波和复数均值滤波不同滤波窗口下最短计算时间(“3×3”窗口下滤波 100 次以内 PSNR 值均未达 88.84,因此不再考虑),结果见表 2。

表 2 表明,对于正余弦均值滤波,每次滤波时重新计算正余弦值和相位值耗时较大,因此滤波次数对滤波时间的影响比滤波窗口影响更大;而对于复数均值滤波,滤波次数与滤波窗口

对总计算时间的影响相当,因此达到相同效果时采用不同滤波窗口的计算时间较为接近。尽管达到同样效果时复数均值滤波所需次数大于正余弦均值滤波,但由于不必每次滤波都重新计算正余弦值和相位值,其计算速度显著快于正余弦均值滤波。

表 2 达到 Butterworth 滤波最佳效果时正余弦均值滤波和复数均值滤波不同滤波窗口下最短计算时间
Tab.2 The shortest times for sin-cos filtering and complex mean filtering with different filtering window sizes to reach the best of Butterworth filtering

算法	PSNR	计算参数	耗时/ms
Butterworth 滤波	88.84	$n=2, D_0=37$	1 416
正余弦均值滤波	88.92	"7×7", 24 次	11 995
正余弦均值滤波	89.00	"5×5", 39 次	15 427
复数均值滤波	89.08	"7×7", 26 次	7 023
复数均值滤波	88.88	"5×5", 41 次	6 719

图 5 为对模拟图进行不同方法滤波所得结果。从图中可以看出,Butterworth 滤波对细节的平滑最为彻底,条纹最为平滑,剩余微小起伏最小,但其缺点在于边缘效应明显,在图像左右边缘出现了平行于边缘的多余条纹,甚至产生了残差点。正余弦均值与复数均值滤波结果基本相同,条纹较不平滑,残余微小起伏较大,边缘效应不明显,滤波图中无残差点。

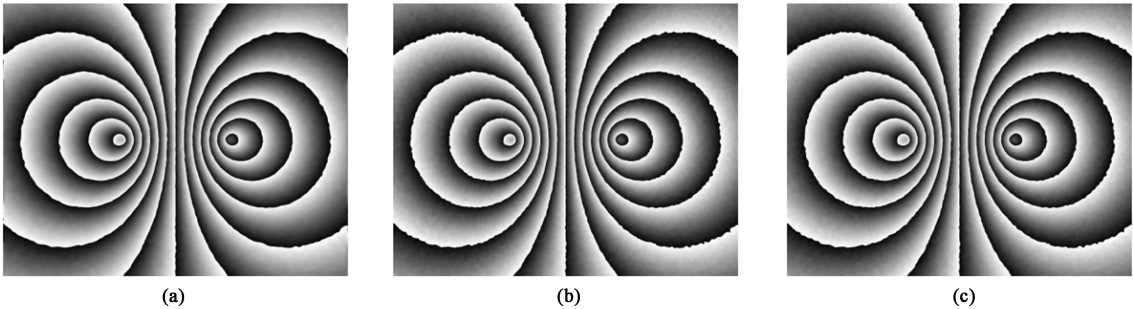


图 5 模拟图滤波结果。(a)Butterworth 滤波, $n=2, D_0=37$; (b)正余弦均值滤波“7×7”,24 次; (c)复数均值滤波,“7×7”,26 次。

Fig.5 Filtered images of simulated diagram with different filtering methods. (a)Butterworth filtering, $n=2, D_0=37$; (b) Sin-cos filtering, “7×7”, 24 times; (c) Complex mean filtering, “7×7”, 26 times.

图 5 表明,Butterworth 滤波对图像内部的滤波效果较好,但边缘部分结果较差;正余弦均值与复数均值滤波对图像内部滤波效果略差,但边缘部分结果较为稳定,滤波效果与图像内部相当。

总体而言,正余弦均值滤波和复数均值滤波达成与之相当的效果所需的计算时间远远超出 Butterworth 滤波。虽然可以通过减少滤波次数缩短计算耗时,但将时间缩短至 Butterworth 滤波耗时将严重恶化滤波效果。因此,在 CPU 平台上,Butterworth 滤波效率远高于正余弦均值滤波和复数均值滤波。

4.2.2 GPU 并行算法滤波

由于 GPU 滤波加速算法基于 CPU 滤波算法,两者结果理论上一致,因此仍按表 2 中的计算参数对各种算法重新测定计算时间。结果如表 3 所示。

表 3 表明,虽然仍然是 Butterworth 滤波速度最快,但其他算法的速度劣势已经大幅减小,可以牺牲一定的精度换取更短的计算时间。例如在 GPU1 上,采用“7×7”窗口进行滤波时,若将滤波次数减少到 15 次,正余弦均值滤波和复数均值滤波需要的时间将缩短为 145 ms 和 161 ms,时间

短于 Butterworth 滤波,相应 PSNR 值为 85.39 和 85.15,与 Butterworth 滤波精度最好时相差不大。就 GPU 加速而言,Butterworth 滤波加速比最低,仅有正余弦均值滤波的 $1/9 \sim 1/6$,复数均值滤波的 $1/3$ 左右。究其原因,主要是因为 Butterworth 滤波需要进行正反快速傅里叶变换,计算中内存读写不连续,控制较为复杂,而正余弦均值滤波和复数均值滤波计算中内存读写连续,有

利于提高内存读写效率。GPU 加速正余弦均值滤波快于复数均值滤波表明,在这两种 GPU 算法中,主要瓶颈已经由不再是正余弦和反正切计算,而是内存的读写,包括线程块内共享内存的读写和线程块与全局显存间的读写。复数均值滤波的全局内存读写量为正余弦均值滤波的 2 倍,因此在消除正余弦和反正切重复计算之后速度仍然慢于正余弦均值滤波。

表 3 采用 GPU 加速计算时,达到 Butterworth 滤波最佳效果时正余弦均值滤波和复数均值滤波不同滤波窗口下最短计算时间

Tab.3 The shortest times for sin-cos filtering and complex mean filtering with different filtering window sizes to reach the best of Butterworth filtering adopting GPU accelerated computing

算法	计算参数	耗时/ms		加速比	
		GPU1	GPU2	GPU1	GPU2
Butterworth 滤波	$n=2, D_0=37$	170	50	8.3	28.3
正余弦均值滤波	" 7×7 ", 24 次	206	71	58	169
正余弦均值滤波	" 5×5 ", 39 次	215	72	72	214
复数均值滤波	" 7×7 ", 26 次	246	80	28.6	88
复数均值滤波	" 5×5 ", 41 次	296	82	22.7	82

4.2.3 解相位差与滤波合并计算

由于滤波读取的数据源自解相位差得到的数据,将相位差数据从 GPU 传到 CPU,保存至硬盘,再从硬盘中读取数据,传到 GPU 进行滤波,可以看出中间传递数据的部分实际上做了无用功。因此,将解相位差与滤波合并计算,即在 GPU 中解相位差后直接进行滤波可进一步提高效率。表 4 为合并解相位差和滤波计算后的总计算时间,其较解相位差全过程时间和单纯滤波时

间之和减少约 $10 \sim 20$ ms,这主要是因为去除了冗余的数据传送过程。

因此,采用 GPU 加速计算后,求解相位差和滤波的总时间可以从 CPU 计算时的 1 900 ms 降低到 239 ms 或 86 ms。采用性能较弱的 GPU 时,可以通过降低滤波次数将正余弦均值滤波时间进一步缩短,从而保证滤波的实时性,如正余弦均值滤波采用" 7×7 "滤波 10 次时,总计算时间为 190 ms,代价是滤波不充分,PSNR 下降至 81.5;

表 4 解相位差合并不同滤波算法的计算总耗时

Tab.4 Total processing time of different filtering method combined with phase calculation

算法	计算参数	耗时/ms	
		GPU1	GPU2
Butterworth 滤波	$n=2, D_0=37$	239	86
正余弦均值滤波	" 7×7 ", 24 次	274	107
正余弦均值滤波	" 5×5 ", 39 次	284	108
复数均值滤波	" 7×7 ", 26 次	315	116
复数均值滤波	" 5×5 ", 41 次	366	116

采用性能较强的 GPU 时,可以直接采用 Butterworth 低通滤波,同时获得较好的滤波效果和滤波实时性。

图 6 为对原始图进行解相位差和滤波所得结果。与模拟图最大的不同在于,因为激光照明质量和遮挡等原因,实际散斑图各处相干程度相差很大。从图 2(相位差图)可以看出,图像中心部位条纹明显,相干程度强,外围条纹暗弱,相干程度弱,图像四角和左侧则无条纹,为非相干区。图 6 表

明,Butterworth 滤波对细节的平滑最为彻底,不仅外围弱相干和非相干区的残差点最少,中心强相干区的微小起伏也最小,但其缺点在于边缘效应明显,在图像边缘平行于边缘的多余条纹。正余弦均值滤波在弱相干区留下的残差点最多,强相干区的微小起伏也最大。复数均值滤波介于两者之间,对弱相干区残差点的消除更接近 Butterworth 滤波,强相干区的微小起伏程度更接近正余弦均值滤波,且和正余弦均值滤波一样边缘效应不明显。

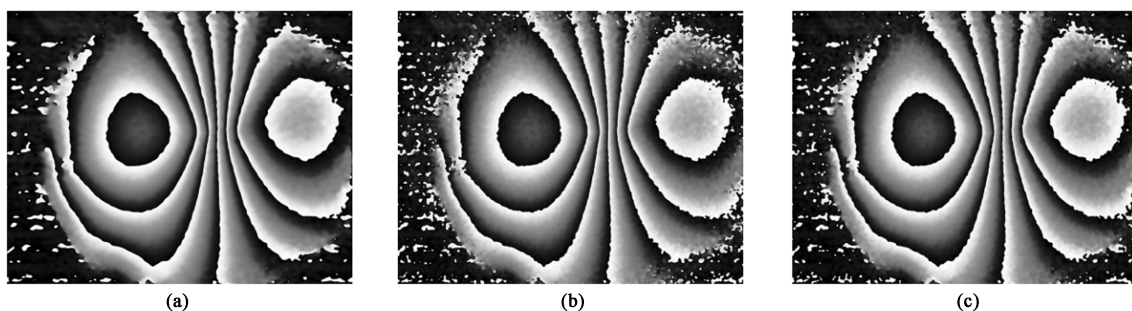


图 6 解相位差和滤波合并计算结果。(a) Butterworth 滤波, $n=2$, $D_0=37$; (b) 正余弦均值滤波“ 7×7 ”, 24 次; (c) 复数均值滤波, “ 7×7 ”, 26 次。

Fig.6 Results of combined phase difference and filtering calculation with different filtering methods: (a) Butterworth filtering, $n=2$, $D_0=37$; (b) Sin-cos filtering, “ 7×7 ”, 24 times; (c) Complex mean filtering, “ 7×7 ”, 26 times.

5 结 论

(1) 采用 GPU 计算可以大幅提高运算速度。相对于 i7-6500U 单线程, GeForce 940MX 的滤波计算速度可以提高 8.3 倍(Butterworth 低通滤波)、58 倍(正余弦均值滤波, “ 7×7 ” 窗口)和 28.6 倍(复数均值滤波, “ 7×7 ” 窗口), Quadro P3000 可提高 28.3 倍(Butterworth 低通滤波)、169 倍(正余弦均值滤波, “ 7×7 ” 窗口)和 88 倍(复数均值滤波, “ 7×7 ” 窗口)。

(2) 不同滤波方法有不同的优缺点, Butterworth 低通滤波对强相干区和弱相干区滤波效果均较好, 滤波结果平滑, 但在图像边缘有严重边缘效应, 易出现异常条纹; 正余弦滤波平滑程度低于 Butterworth 低通滤波, 在弱相干区容易残留大

量残差点, 而边缘效应不明显; 复数均值滤波平滑程度与正余弦滤波相似, 在弱相干区效果与 Butterworth 低通滤波相似, 边缘效应不明显。

(3) 采用 GPU 计算时, 速度瓶颈很大程度上出现在数据传输中, 包括系统内存与 GPU 显存之间的传输和 GPU 核心与显存间的传输。数据传输量越少、显存读写连续性越好, 越有利于发挥 GPU 计算速度快的优点。

(4) 对于性能不同的 GPU, 可以采取不同的算法在滤波速度和滤波精度间取得平衡。对性能较低的 GPU, 可以适当降低精度要求, 采用正余弦均值滤波通过减少滤波次数的方式提高滤波速度, 保证实时性; 对于性能较强的 GPU, 可以采用 Butterworth 低通滤波同时获得较好的滤波效果和滤波实时性; 如果 GPU 性能更强, 可以采用正余弦均值滤波增加滤波次数以获得更好的滤波效果。

参 考 文 献:

- [1] RASTOGI P K. Speckle and speckle shearing interferometry—II. At the threshold of a new era [J]. *Optics and*

- Lasers in Engineering*, 1997, 26(4/5): 279-282.
- [2] 洪友仁,何浩培,何小元.剪切散斑:一种光学测量技术及其应用[J].实验力学,2006,21(6):667-688.
HUNGY Y, HO H P, HE X Y. Shearography: an optical measurement technique and applications [J]. *Journal of Experimental Mechanics*, 2006, 21(6): 667-688. (in Chinese)
- [3] WANG Y H, THOMAS D, ZHANG P, *et al.* Whole field strain measurement on complex surfaces by digital speckle pattern interferometry [J]. *Materials Evaluation*, 2008, 66(5): 507-512.
- [4] 李峰.电子散斑干涉的图像处理及涡旋光特性研究[D].济南:山东师范大学,2015.
LI F. Image processing of electronic speckle pattern interference and optical characteristics research of vortex beam [D]. Ji'nan: Shandong Normal University, 2015. (in Chinese)
- [5] 宋凡峰,王开福,郑蓝.电子散斑相位检测的滤波方法研究[J].激光杂志,2007,28(3):50-51.
SONG F F, WANG K F, ZHENG L. Investigation of filtering method for phase measurement in electronic speckle pattern interferometry (ESPI) [J]. *Laser Journal*, 2008, 28(3): 50-51. (in Chinese)
- [6] KEMAO Q, NAM L T H, FENG L, *et al.* Comparative analysis on some filters for wrapped phase maps [J]. *Applied Optics*, 2007, 46(30): 7412-7418.
- [7] 李灿灿.电子散斑干涉图像滤波方法研究及在动态热变形中的应用[D].天津:天津大学,2012.
LI C C. Study of electronic speckle pattern interferometry and its application on dynamic thermal deformation [D]. Tianjin: Tianjin University, 2012. (in Chinese)
- [8] 顾国庆,王开福,燕新九.基于同态滤波的电子散斑干涉图像处理[J].激光技术,2010,34(6):750-752,797.
GU G Q, WANG K F, YAN X J. Electronic speckle interferometry image processing based on homomorphic filtering [J]. *Laser Technology*, 2010, 34(6): 750-752, 797. (in Chinese)
- [9] HA S, PARK J, YOU D. A GPU-accelerated semi-implicit fractional-step method for numerical solutions of incompressible Navier - Stokes equations [J]. *Journal of Computational Physics*, 2018, 352: 246-264.
- [10] WINKLER D, MEISTER M, REZAVAND M, *et al.* gpuSPHASE—A shared memory caching implementation for 2D SPH using CUDA [J]. *Computer Physics Communications*, 2017, 213: 165-180.
- [11] SOLTANI H, LY D Q, AHMED W. Accelerating cell dynamics simulations of soft materials using CUDA-GPU [J]. *Materials Today: Proceedings*, 2017, 4(1): 75-86.
- [12] LIU W G, SCHMIDT B, VOSS G. Accelerating molecular dynamics simulations using graphics processing units with CUDA [J]. *Computer Physics Communications*, 2008, 179(9): 634-641.
- [13] GONG T, FAN T T, GUO J Z, *et al.* GPU-based parallel optimization of immune convolutional neural network and embedded system [J]. *Engineering Applications of Artificial Intelligence*, 2017, 62: 384-395.
- [14] FONSECA A, CABRAL B. Prototyping a GPGPU neural network for deep-learning big data analysis [J]. *Big Data Research*, 2017, 8: 50-56.
- [15] LITJENS G, KOOI T, BEJNORDI B E, *et al.* A survey on deep learning in medical image analysis [J]. *Medical Image Analysis*, 2017, 42: 60-88.
- [16] LOCANS U, ADELMANN A, SUTER A, *et al.* Real-time computation of parameter fitting and image reconstruction using graphical processing units [J]. *Computer Physics Communications*, 2017, 215: 71-80.
- [17] PINHEIRO A, DESTERRO F, SANTOS M C, *et al.* GPU-based implementation of a diagnostic wind field model used in real-time prediction of atmospheric dispersion of radionuclides [J]. *Progress in Nuclear Energy*, 2017, 100: 146-163.
- [18] MAAS A A M, SOMERS P A A M. Two-dimensional deconvolution applied to phase-stepped shearography [J]. *Optics and Lasers in Engineering*, 1997, 26(4/5): 351-360.

作者简介:



邵珩(1989—),男,湖北鄂州人,博士,工程师,2017年于清华大学获得博士学位,主要从事激光干涉测量及图像处理等方面的研究。E-mail: shao_heng@126.com