《光电成像与测量综合技术》专题文章导读

张　葆
中国科学院 长春光学精密机械与物理研究所航空成像与测量技术研究部

近年来，随着现代军事技术发展的需求，侦察情报在战争中的地位愈显突出。由于固定站的侦察视野受光学系统视场角的限制，获得的侦察信息相对较少，现代战争逐渐把侦察系统安放在移动站（地面车辆、舰船、飞机、卫星等动载体）上来扩大光学系统的动态视场并增加收容信息，弥补固定站的不足。但是，随之而来的是动载体振动、大气扰动、速度、传感器、光学系统、平台精度、系统集成、可靠性等对成像质量的威胁和影响。因此，有必要关注和研究影响成像质量的各个环节，并采取相应的针对性措施。

本课题组针对上述问题展开了若干专题性研究，旨在通过这些研究提升侦察装备的性能并拓展其功能，从而提高动载体成像系统的清晰度与分辨能力，提高成像系统的图像质量、侦察效果，同时拓展其应用空间。

本文是长期从事光电侦察领域的技术人员结合理论及工作需求，总结的一些提高光电成像质量的综合技术：文一《基于图形处理器的实数 FFT 在图像处理中的应用》，介绍了一种图像恢复技术，解决了光电成像系统视频图像帧内运动模糊的快速恢复问题；文二《非均匀性校正在红外杂散辐射抑制中的应用》，利用非均匀性校正技术抑制红外杂散辐射，提高了红外系统的成像质量；文三《气象测云红外成像系统的设计与分析》，全面分析了红外成像系统的关键技术；文四《机载光电成像设备的可测试性系统设计》，详细叙述了机载光电成像设备的可测试性系统设计的过程；文五《机载光电稳定平台框架结构工程分析》，通过工程分析手段有效地减轻平台重量并保证了结构固有频率。

希望上述文章能对从事动载体光电成像领域的技术人员有所帮助，并提出宝贵意见。

文章编号　1004-924X(2008)12-2414-07

基于图形处理器的实数 FFT 在图像处理中的应用

李　仕[1,2]，王　晶[1]，孙　辉[1]
（1. 中国科学院 长春光学精密机械与物理研究所，吉林 长春 130033；
2. 中国科学院 研究生院，北京 100039）

摘要：对常用的实数快速傅里叶变换（FFT）算法进行改进，在此基础上提出基于图形处理器（GPU）平台的实数 FFT 算法。常用的实数 FFT 算法比传统的复数 FFT 算法的运算量降低了约 40%，文中提出的实数 FFT 算法的运算量能减少到复数 FFT 算法的 50%。通过对存储数据的特殊配置，改进后的实数 FFT 算法能由一维拓展到二维，并通过调用 CUFFT 库的一维复数 FFT 函数，实现实数 FFT 算法的并行运算。实验中通过维纳滤波算法的应用，对实数 FFT 算法的精确度及效率进行检验。图像处理效果表明，文中实数 FFT 算法是正确、可行的，可以帮助维纳滤波算法在 19.26 ms 内恢复一帧 2 048×2 048 灰度模糊图像，该速度是在 GPU 上使用复数 FFT 维纳滤波的 2.34 倍，是在 CPU 上使用实数 FFT 维纳滤波的 37.46 倍。实验表明文中算法具有一定的实用价值。

关　键　词：实数快速傅里叶变换；图像恢复；图形处理器；实时运算
中图分类号：TP391.4　　**文献标识码**：A

收稿日期：2008-08-06；**修订日期**：2008-10-14.
基金项目：国防科技预研基金资助项目（No. 1040603）

Real FFT based on graphic processing unit for image processing

LI Shi[1,2], WANG Jing[1], SUN Hui[1]

(1. *Changchun Institute of Optics, Fine Mechanics and Physics, Chinese Academy of Sciences, Changchun* 130033, *China*;
2. *Graduate University of Chinese Academy of Sciences, Beijing* 100039, *China*)

Abstract: By analysis of general real Fast Fourier Transform (FFT) algorithm, an improved real FFT algorithm is presented based on Graphic Processing Unit (GPU). A general real FFT can reduce the computation consumption of the conventional complex FFT algorithm by about 40%, and the presented real FFT algorithm improves the reduction to 50%. On the basis of the specific configuration of data storage, the presented real FFT algorithm can be applied in two directions continuously to generate the 2D Fourier spectrum, also the parallel real FFT transform can be implemented by using the 1D complex FFT function from the CUFFT library. The precision and the efficiency of the presented real FFT are inspected by Wiener filter's application, image processing results demonstrate that the presented real FFT algorithm shows operating properly and working well. With the help of the presented real FFT, Wiener filter can restore a 2 048×2 048 8-bit image by motion speed of 19.26 ms, which is 2.34 times as compared with that of the Wiener filter based on complex FFT on GPU, and is 37.46 times that of the Wiener filter based on real FFT on CPU. The experiment results show that the presented algorithm is practical for image processing.

Key words: real Fast Fourier Transform(FFT); image restoration; Graphic Processing Unit(GPU); real-time computing

1　引　言

1965 年，J. W. Cooley 和 J. W. Tukey 提出快速傅立叶变换[1](FFT)，使得空间域的卷积(去卷积)运算可被快速地转换为频域的乘法(除法)运算，这在信号处理领域，尤其在图像处理领域产生了革命性的影响。FFT 算法的运算效率并没有达到最优化，Glenn D. Bergland 在 1968 年最早提出了实数 FFT[2]，使得 FFT 运算中多余的运算量及存储空间得到有效压缩，在前期工作中本课题组验证了这点[3]。关于 FFT 算法的研究并没就此停止，2003 年 Kenneth Moreland 与 Edward Angel 开始在 GPU 平台上移植 FFT 算法[4]，虽然他们研究的出发点是为了挖掘 GPU 处理器的通用计算潜力，但从此在全球范围掀起一股将 GPU 用于通用计算(GPGPU)[5]的研究热潮。现在每隔半年左右新一代 GPU 便会诞生，Super Computing2007 大会上 nVidia 宣称 GPU 的计算性能大约是同期 CPU 的 10 倍。2007 年 NVidia 公司推出基于 GPU 并行运算的 FFT 库函数 CUFFT，使得实数 FFT 算法相对复数 FFT 所带来的效率提升显得微不足道。

在前期工作使用 CUFFT 库函数的过程发现，在 GPU 上执行复数 FFT 的效率是在同档 CPU 上运行实数 FFT 算法效率的 20 倍。像一般的复数 FFT 一样，在使用 CUFFT 库函数时，需通过填零，将图像的实数数据填充成复数数据。这必将使得 FFT 算法在存储空间与运算量上存在冗余[6]。CUFFT 库里面有用于实数运算的 FFT，但 CUFFT 所提供的实数 FFT 函数实质上还是复数傅立叶变换，它只是用传统实数 FFT 的接口形式对 CUFFT 中的复数 FFT 进行封装，并没有提高 FFT 函数在 GPU 上的运算效率。

CUFFT 库函数的高效性是有目共睹的，但 GPU 数据存储区大小是有限的，CUFFT 用的复数 FFT 算法所产生的存储空间浪费直接影响到 GPU 所能处理的最大图像的尺寸。为此，本文探讨如何在 CUFFT 的基础上实现实数 FFT 算法，并以此进一步提高图像频域处理算法的运算效率和 GPU 硬件资源的使用效率。

2　实数 FFT

2.1　频谱特性

FFT 变换中的多维 FFT 变换可以通过一维 FFT 的重复运算来获得，这里仅讨论一维 FFT 的情形。一维 FFT 公式如(1)式所示：

$$F(u)=\sum_{x=0}^{N-1}f(x)W_N^{ux}, \tag{1}$$

式中 N 表示复数序列 $f(x)$ 的个数，$W_N=e^{-i2\pi/N}$。当 $f(x)$ 的虚部全部为“0”的时候，由式(1)能得到如式(2)所示的共轭关系，并可由此得到共轭对称频谱图。

$$F(u)=F^*(N-u). \tag{2}$$

2.2　实数 FFT 构建

常用的实数 FFT 计算方法是将 N 点的实数序列 $f(x)$ 进行奇偶分离，并令 $f^e(k)=f(2x)$，$f^o(k)=f(2x+1)$，按照式(3)所示构建 $N/2$ 点复数序列 $h(k)$。

$$h(k)=f^e(k)+if^o(k). \tag{3}$$

根据 FFT 的线性运算性质，得到如下频谱关系：

$$H(u)=F^e(u)+iF^o(u). \tag{4}$$

2.3　频谱分解

由于 $f^e(k)$ 与 $f^o(k)$ 均为实数，所以 $F^e(u)$ 与 $F^o(u)$ 会呈式(2)所示的共轭对称关系。即：

$$H(N-u)=F^e(N-u)+iF^o(N-u)=F^{e*}(u)+iF^{o*}(u). \tag{5}$$

由式(4)、(5)知

$$\begin{cases}H(u)_R=F^e(u)_R-F^o(u)_I\\H(u)_T=F^e(u)_I+F^o(u)_R\\H(N-u)_R=F^e(u)_R+F^o(u)_I\\H(N-u)_I=F^o(u)_R-F^e(u)_I\end{cases}. \tag{6}$$

由式(6)，分别求得

$$\begin{cases}F^e(u)_R=\frac{1}{2}[H(u)_R+H(N-u)_R]\\F^e(u)_I=\frac{1}{2}[H(u)_I-H(N-u)_I]\\F^o(u)_R=\frac{1}{2}[H(u)_I+H(N-u)_I]\\F^o(u)_I=\frac{1}{2}[H(N-u)_R-H(u)_R]\end{cases}. \tag{7}$$

如无特别说明，文中的下标 R、I 分别表示相应复数的实部和虚部。

2.4　频谱合成

由式(1)可得到如下推导：

$$F(u)=\sum_{x=0}^{N-1}f(x)W_N^{xu}=\sum_{k=0}^{N/2-1}f^e(k)W_N^{2ku}+\sum_{k=0}^{N/2-1}f^o(k)W_N^{(2k+1)u}=\sum_{k=0}^{N/2-1}f^e(k)W_N^{2ku}+W_N^u\sum_{k=0}^{N/2-1}f^o(k)W_N^{2ku}. \tag{8}$$

由于，$W_N^2=e^{(-i2\pi/N)2}=e^{-i2\pi/(N/2)}=W_{N/2}$，代入式(8)得：

$$F(u)=\sum_{k=0}^{N/2-1}f^e(k)W_{N/2}^{ku}+W_N^u\sum_{k=0}^{N/2-1}f^o(k)W_{N/2}^{ku}=F^e(u)+W_N^uF^o(u), \tag{9}$$

即按照式(9)对式(7)的结果进行蝶形运算，便能获得实数序列 $f(x)$ 的频谱 $F(u)$。通过以上推导，原来需要 N 点复数 FFT 的运算被缩减为 $N/2$点复数 FFT 的运算，然后通过 2.3 节的频谱分解及 2.4 节的频谱合成，生成原来 N 点实数序列的频谱，该算法的运算量大约是直接进行复数 FFT 的运算量的 60%。

3　实数 FFT 在 GPU 上的并行方案

3.1　一维实数 FFT 的改进

前一节介绍的实数 FFT 算法虽然能将原运算量降低 40%，但是该算法的运算复杂度相对原复数 FFT 算法有所增加，不利于在 GPU 上编程实现。针对图像处理这一特殊应用对象，现对前面算法进行改进。

设 $f_k(x)$ 为图像上的第 k 行像素序列，用如

下方式构建复数序列：

$$h(x)=f_{2k}(x)+\mathrm{i}f_{2k+1}(x)\ . \tag{10}$$

通过 FFT 的线性运算性质得到：

$$H(u)=F_{2k}(u)+\mathrm{i}F_{2k+1}(u)\ , \tag{11}$$

根据 2.3 节的推导，可求得：

$$\begin{aligned}
F_{2k}(u)_{\mathrm{R}}&=\frac{1}{2}[H(u)_{\mathrm{R}}+H(N-u)_{\mathrm{R}}]\\
F_{2k}(u)_{\mathrm{I}}&=\frac{1}{2}[H(u)_{\mathrm{I}}-H(N-u)_{\mathrm{I}}]\\
F_{2k+1}(u)_{\mathrm{R}}&=\frac{1}{2}[H(u)_{\mathrm{I}}+H(N-u)_{\mathrm{I}}]\\
F_{2k+1}(u)_{\mathrm{I}}&=\frac{1}{2}[H(N-u)_{\mathrm{R}}-H(u)_{\mathrm{R}}]
\end{aligned}\ . \tag{12}$$

由式(12)可发现用改进的方法构建复数序列，在经过复数 FFT 运算后只需再进行简单的加减运算和位运算就能得到原实数序列 $f_{2k}(x)$、$f_{2k+1}(x)$的对应频谱 $F_{2k}(u)$、$F_{2k+1}(u)$。改进后的算法避免了复杂的频谱合成运算，提高了实数 FFT 运算的效率，它的运算量大约是直接进行复数 FFT 运算的 50%。

3.2　一维实数 FFT 的数据配置

3.1 节的方法可以直接应用在文献[7]中的基于一维频谱的图像恢复算法上，但如果要应用到基于二维频谱的算法上时，则还需进一步推导二维实数 FFT 算法。3.1 节中一维实数 FFT 的存储配置为：

进行实数 FFT 运算前的数据存储如图 1(a)所示，第 $2k$ 行的数据被依次存放在新复数序列的实部位置，第 $2k+1$ 行的数据按顺序存储在新复数序列的虚部位置。对新复数序列进行复数 FFT 运算，并经式(12)的简单运算得到图 1(b)所示的频谱。由式(12)知道 $F(0)_{\mathrm{I}}$、$F(N/2)_{\mathrm{I}}$ 均为零值，为节省存储空间，将 $F(0)_{\mathrm{R}}$、$F(N/2)_{\mathrm{R}}$ 两数作为一个特殊复数进行存储(见图 1(b))。通过式(2)所示的共轭对称关系，图 1(b)所示的频谱图能很容易地被展开为第 $2k$ 行和第 $2k+1$ 行数据各自完整的频谱 $F_{2k}(u)$ 和 $F_{2k+1}(u)$。数字数据都是按线性进行存储，二维图像数据实际上就是采用一维的线性存储方式进行存储。为方便描述，将图 1(b)中的一维数据改写成图 1(c)所示的二维数据形式。

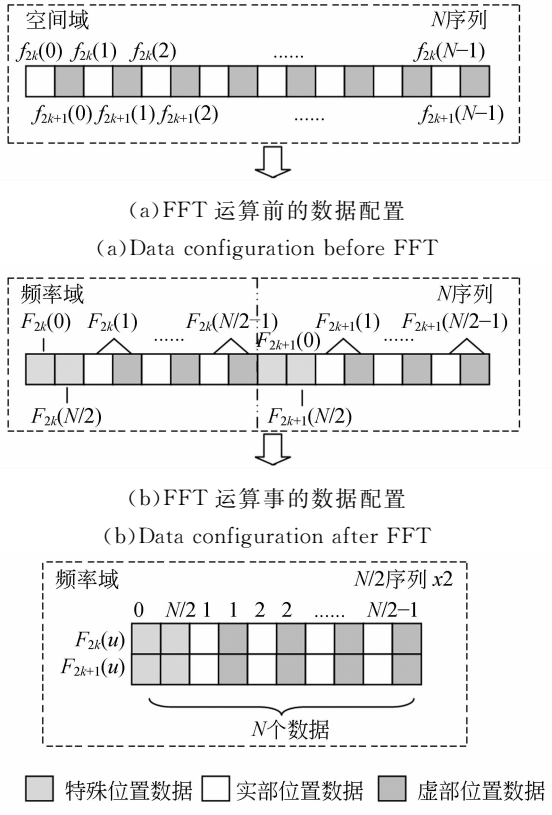


图 1　一维实数 FFT 的数据存储设置

Fig. 1　Data organization of 1D real FFT

3.3　二维实数 FFT 的拓展

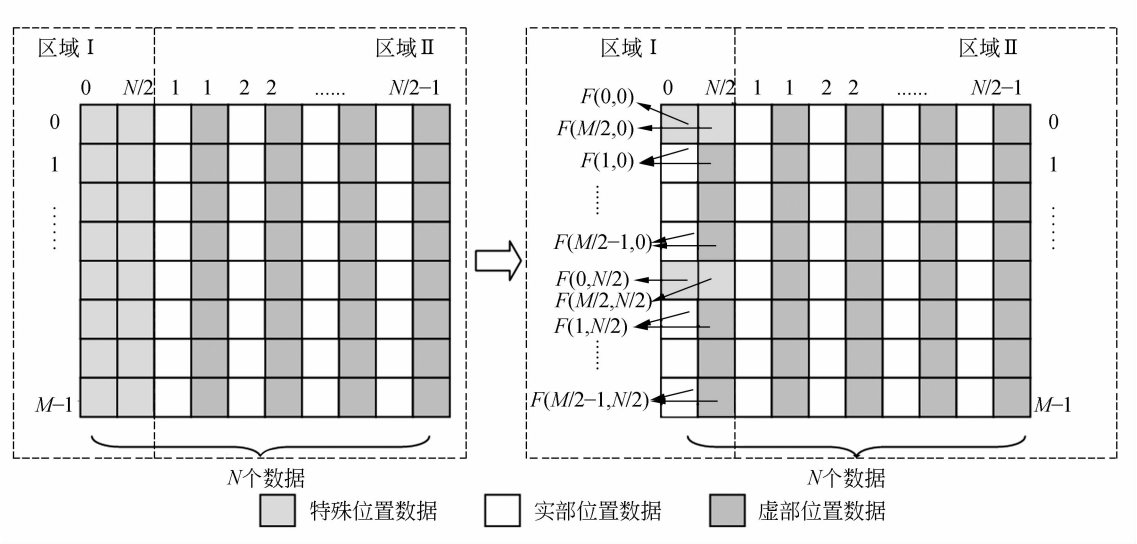
二维图像数据的二维 FFT 频谱可以通过对图像数据分别进行行方向上一维 FFT 和列方向上一维 FFT 得到。由图 1(c)可推得二维图像数据经行方向上一维实数 FFT 运算后的频谱图(如图 2(a)所示)。因此，对图 2(a)所示的频谱再进行一次列方向上的一维 FFT 运算便能得到原图像的二维 FFT 频谱。

图 2(a)中区域Ⅱ内的$(N/2-1)\times 2$ 列的数据，对应为$(N/2-1)$列复数序列，对该$(N/2-1)$列复数序列分别做一维复数 FFT 运算，所得结果(图 2(b)中区域Ⅱ内数据)即为图像二维复数 FFT 频谱的半频谱。通过频谱的共轭对称关系，不难将半频谱展开为完整频谱。

图 2(a)中区域Ⅰ内的两列数据为两列实数序列，分别对应每行数据在一维 FFT 运算后第 0 点和第 $N/2$ 点的频谱。通过 3.1 节所述的方法，能求得这两列数据各自在列方向上的频谱，所得结果(图 2(b)中区域Ⅰ内数据)分别为二维复数

FFT 频谱中 0 列和 $N/2$ 列上的对应数据。

通过对二维图像数据做行方向上的一维实数 FFT 运算，得到频谱图 2(a)，再对图 2(a)频谱数据进行列方向上的一维复数 FFT 运算，得到图 2(b)所示的频谱。对图 2(b)频谱按照共轭关系展开便能得到图像的完整二维频谱。整个运算过程，可以直接调用 CUFFT 库函数中的一维复数 FFT 函数来进行本文的实数 FFT 运算，借助 CUFFT 库函数及 GPU 的并行运算特点，实现实数 FFT 的并行处理。



(a)一维 FFT 运算后的图像数据配置　(b)二维实数 FFT 的数据配置

(a)Image data configuration after 1D FFT　(b)Data configuration of 2D real FFT

图 2　二维实数 FFT 的数据存储设置

Fig. 2　Data organizations of 2D real FFT

4　实验结果

本文实数 FFT 算法的提出是为了提高图像算法的运行速度。由于单独的实数 FFT 运算较难做算法精度上的检验[6]，因此在实验中用维纳滤波算法[8-9]作为载体来对文中的二维实数 FFT 算法进行精度及运算效率的评测。实验测试平台的 GPU 为 GeForce8800GTS(显存 500M)，CPU 是 P4 主频 3.0 G(单核)，内存 1 G。测试程序在 VC2005 环境下编译完成。图 3 为实验的原始图像，图 4 为人工卷积模糊后的图像，图 5 为经维纳滤波算法恢复所得的图像。从图 4 的图像卷积模糊的效果及图 5 的图像去卷积模糊的效果可看出，文中所述的二维实数 FFT 算法能替代复数 FFT 进行工作。表 1 的评测数据是以图 3 为参照分别对图 4 及图 5 的图像质量进行测试所得，其结果与前期工作[3,7]的结果基本一致。

图 3　原图 (512×512)

Fig. 3　Original image (512×512)

图 4　模糊图像 (512×512)

Fig. 4　Blurred image (512×512)

图 5　复原图像 (512×512)

Fig. 5　Restored image (512×512)

表 1　图像恢复前后 MSE、PSNR 值对照

Tab. 1　MSE and PSNR fore-and-aft restoration

图像	尺寸	MSE	PSNR
模糊(起居室)	512×512	638.72	20.08
恢复(起居室)	512×512	35.96	32.57

表 2 数据为分别将基于 CPU 运算的实数 FFT 算法(CPU_R_FFT)、基于 GPU 运算的复数 FFT 算法(GPU_C_FFT)及基于 GPU 运算的实数 FFT 算法(GPU_R_FFT)用于维纳滤波算法中,并对各不同分辨率的图像进行图像恢复实验所得的数据。表 2 数据表明文中所提的实数 FFT 算法相对原 CUFFT 中的复数 FFT 算法效率提升明显,尤其对 2 048×2 048 分辨率的图像,效率达到原先的 2.34 倍。而当维纳滤波使用基于 GPU 的实数 FFT 时,维纳滤波的处理速度是使用基于 CPU 的实数 FFT 时的 40 倍左右。表 2 数据同时显示本文算法很好地实现了 2 048×2 048灰度图像的实时恢复。

表 2　耗时测试结果(单位:ms)

Tab. 2　Results of time consumption(unit: ms)

图像大小	256×256	512×512	1 024×1 024	2 048×2 048
CPU_R_FFT①	6.93	40.31	165.96	721.53
GPU_C_FFT②	0.467	2.05	7.93	45.01
GPU_R_FFT③	0.272	0.98	3.98	19.26
②/③	1.72x	2.09x	1.99x	2.34x
①/③	25.48x	41.13x	41.70x	37.46x

5　结　论

本文在一般的实数 FFT 算法的基础上进行改进,使得实数 FFT 算法的运算量从之前 60% 的运算量降低到 50%的运算量,算法效率在原基础上提升 20%。实验通过调用 CUFFT 中一维复数 FFT 函数来实现一维及二维的实数 FFT 算法。GPU 编程的并行运算特性,及 CUFFT 本身针对 GPU 硬件的高度优化,使得本文算法在执行时候间接具有并行运算的能力。实验中维纳滤波算法的运算效果表明本文的实数 FFT 算法能达到一定的算法精度,且能满足正常的图像处理应用需求。文中实数 FFT 算法在不同处理平台的纵向对比和同一平台的横向对比中,均展现出高效的运算性能。其中使得单帧 2 048×2 048 灰度模糊图像在 19.26 ms 内得到恢复这一实验结果在工程应用上具有深远意义。

参考文献:

[1] COOLEY J W,TUKEY J W. An algorithm for the machine calculation of complex Fourier series[J]. *Mathematics Computation*, 1965,19 :296-301.

[2] BERGLAND G D. A fast fourier transform algorithm for real-valued series[J]. *Comm. of the ACM*, 1968, 11 (10): 703-710.

[3] 李仕,张葆,孙辉.航空光电成像模糊的实时恢复[J].光学 精密工程,2007,15(8):1287-1292.
LI SH, ZHANG B, SUN H. Real-time restoration using real discrete Fourier transform for aerial E-O imaging system [J]. *Opt. Precision Eng.*, 2007, 15(8): 1287-1292. (in Chinese)

[4] MORELAND K, ANGEL E. The FFT on a GPU[C]. *SIGGRAPH/Eurographics Workshop on Graphics Hardware* 2003 *Proceedings*, 2003, 7: 112-119.

[5] PHARR M, FERNANDO R. *GPU Gems*2[M]. Boston: Addison-Wesley Professional, 2005.

[6] SMITH S W. *The Scientist and Engineer's Guide to Digital Signal Processing*[M]. San Diego: California Technical Publishing, 1997.

[7] 李仕,孙辉,张葆.一种运动模糊图像的实时恢复算法[J].光学 精密工程,2007,15(5):767-772.
LI SH, SUN H, ZHANG B. A method for real-time restoration of motion-blurred images [J]. *Opt. Precision Eng.*, 2007, 15(5): 767-772. (in Chinese)

[8] ANDREWS H C, HUNT B R. *Digital Image Restoration*[M]. Englewood Cliffs, NJ: Prentice-Hall, 1977.

[9] 贾平,张葆,孙辉. 航空成像像移模糊恢复技术[J].光学 精密工程,2006,14(4):697-703.
JIA P, ZHANG B, SUN H. Restoration of motion-blurred aerial image [J]. *Opt. Precision Eng.*, 2006, 14(4): 697-703. (in Chinese)

作者简介:李仕(1984－),男,浙江苍南人,博士研究生,主要从事航空成像补偿算法的研究。E-mail: brightlishi@gmail.com

通讯作者:王晶(1966－),女,吉林九台人,研究员,主要从事光电成像与测量技术的研究。E-mail: cleresky@vip.sina.com

孙辉(1963－),男,吉林柳河人,研究员,主要研究方向为数字图像处理与分析、计算机仿真技术等。E-mail: sunh@ciomp.ac.cn

●下期预告

空间太阳望远镜中的轻量化铍镜研究

宋立强[1,2],杨世模[1],陈志远[1]

(1.中国科学院 国家天文台,北京 100012;2.中国科学院 研究生院,北京 100049)

为提高空间太阳望远镜相关跟踪系统的动态性能、验证金属铍作为工作在可见光波段反射镜材料的可行性与可靠性,进行了空间太阳望远镜相关跟踪器中的摆镜研究,开展了 Φ84 mm 轻量化铍镜设计与研制。利用冲击研磨工艺得到铍粉,再通过热等静压工艺和机械加工得到铍镜镜坯。使用化学镀镍工艺在镜坯基体上镀覆镍磷合金过渡层,再经过光学加工完成铍镜研制。镜面干涉检测后得到面形精度,RMS:0.012 λ,PV:0.114 λ,铍镜轻量化率为 43.68%。检测结果满足空间太阳望远镜的技术要求。此铍镜的研制完成,不仅突破了高精度铍镜研制的技术路线,也表明金属铍是空间天文仪器中优异的反射镜基体材料,为在我国空间天文仪器中应用铍镜奠定了基础。