DOI:10.13203/j.whugis20180248 文章编号:1671-8860(2018)12-1838-09

基于 FPGA 的 P-H 法星上解算卫星相对姿态

周国清[1,2] 黄景金[1] 舒 磊[1]

1 天津大学精密仪器与光电子工程学院,天津,300072
2 桂林理工大学广西空间信息与测绘重点实验室,广西 桂林,541004

摘 要:针对目前星上遥感图像实时处理只能实现低级别算法的情况,提出了基于现场可编程门阵列(field-programmable gate array,FPGA)的 P-H 法星上相对姿态实时解算模型。该模型不仅避免了传统基于欧拉角的复杂三角函数计算与初值估算,还降低了迭代次数。试验选用 FPGA(V7 xc7vx1140t)作为实时解算的硬件平台。在 FPGA 实现中,采用 64 位的浮点数据结构和串行/并行相结合策略;并采用 LU(Lower-Upper)分解-分块算法实现矩阵求逆。试验结果表明,该模型的迭代次数比基于欧拉角的少了 13 次。该模型在 FPGA 和计算机的实现结果相差仅为 5.0×10^{-14},加速度比为 10。另外,该模型可广泛适用于实时性要求高的图像处理领域。

关键词:P-H 法;单位四元数;相对姿态;LU 分解-分块算法;FPGA

中图分类号:P23 **文献标志码**:A

随着各方面对卫星产品的多样性、实时性等需求不断提高,特别是军事行动、应急救灾等方面,国内外相继提出了智能地球观测卫星系统[1-3]、对地观测脑[4]等概念,这些系统都属于从数字地球发展到智慧地球的必然过程[5]。这些系统具有基于事件驱动,遥感数据星上实时处理,遥感产品实时分发的特点。而遥感数据具有空间、时间、光谱分辨率高等优点,但遥感卫星硬件资源有限、功耗低、体积小且实时性要求高,因此,遥感数据的星上实时处理是亟需解决的核心技术之一。

针对星上实时处理的特点,一般采用基于现场可编程门阵列(field-programmable gate array,FPGA)[3-4]或者 FPGA+数字信号处理(digital signal processing,DSP)[6]作为硬件架构。基于上述架构,目前能够实现星上实时处理的算法有分类[7]、数据压缩[8]、云检测[9-10]、图像复原[11]等。这类算法均属于遥感图像的预处理阶段(低级别算法),大部分算法采用定点数据结构即可完成。针对计算量大、时间复杂度高、浮点数据结构的算法,如三维重建、目标跟踪与识别、姿态解算等,目前还无法在星上进行实时处理。

笔者在基于 FPGA 的局部特征检测与匹配[12-13]和基于 FPGA 的几何校正[14]的基础上,提出卫星相对姿态解算的 FPGA 实现。主要创新有:①在算法方面,采用单位四元数的 P-H 法相对姿态解算模型,该模型避免了旋转矩阵中大量三角函数的计算,同时减少了迭代次数,提高了迭代稳定性[15-16];②在矩阵求逆方面,直接的矩阵求逆方法会消耗大量的硬件资源和加大时间延迟。虽然传统的 LU(lower-upper)分解算法降低了矩阵求逆的时间复杂度,但由于矩阵元素之间存在严重的依赖关系,并行度不大,在 FPGA 上实现难度较大。为解决上述问题,本文提出基于 FPGA 的 LU 分解-分块算法。该算法不仅降低了矩阵的维度和时间复杂度,更提高了算法本身的并行特性,适用于具有并行特性的 FPGA 硬件平台;③在数据结构和实现策略方面,为保证计算精度,采用了 64 位的双精度浮点数据结构和串/并行相结合的实现策略。这 3 个创新点保证了迭代结果精度高,处理速度快,硬件资源消耗少。

1 P-H 法相对姿态解算模型

1.1 P-H 法定义[17-18]

令 $q=d+\mathrm{i}a+\mathrm{j}b+\mathrm{k}c$ 是单位四元数(i、j、k 是

收稿日期:2018-08-02

项目资助:国家自然科学基金(41431179);国家重点研发计划(2016YFB0502501);广西创新驱动发展专项(科技重大专项)(桂科 AA18118038)。

第一作者:周国清,博士,教授,主要从事智能卫星观测系统、激光雷达研究。gzhou@glut.edu.cn

通讯作者:黄景金,博士生。jingjin_huang@tju.edu.cn

虚数单位,d、a、b、c 为实数,且 $d^2+a^2+b^2+c^2=1$)。旋转矩阵 $\boldsymbol{R}$ 的四元数表达式为:

$$\boldsymbol{R}=\begin{bmatrix} d^2+a^2-b^2-c^2 & 2(ab-cd) & 2(ac+bd) \\ 2(ab+cd) & d^2-a^2+b^2-c^2 & 2(bc-ad) \\ 2(ac-bd) & 2(bc+ad) & d^2-a^2-b^2+c^2 \end{bmatrix} \tag{1}$$

$\boldsymbol{R}$ 具有正交性,即 $\boldsymbol{R}\boldsymbol{R}^{\mathrm{T}}=\boldsymbol{I}$,求微分可得:

$$\mathrm{d}(\boldsymbol{R}\boldsymbol{R}^{\mathrm{T}})=\mathrm{d}\boldsymbol{R}\boldsymbol{R}^{\mathrm{T}}+\boldsymbol{R}\mathrm{d}\boldsymbol{R}^{\mathrm{T}}=0 \tag{2}$$

变换为:

$$\mathrm{d}\boldsymbol{R}\boldsymbol{R}^{\mathrm{T}}=-\boldsymbol{R}\mathrm{d}\boldsymbol{R}^{\mathrm{T}}=-(\mathrm{d}\boldsymbol{R}\boldsymbol{R}^{\mathrm{T}})^{\mathrm{T}} \tag{3}$$

式(3)为反对称阵,令:

$$\boldsymbol{S}_w=\mathrm{d}\boldsymbol{R}\boldsymbol{R}^{\mathrm{T}} \tag{4}$$

则:

$$\boldsymbol{S}_w\boldsymbol{R}=\mathrm{d}\boldsymbol{R}\boldsymbol{R}^{\mathrm{T}}\boldsymbol{R} \tag{5}$$

又因 $\boldsymbol{R}^{-1}=\boldsymbol{R}^{\mathrm{T}}$,得:

$$\boldsymbol{S}_w\boldsymbol{R}=\mathrm{d}\boldsymbol{R} \tag{6}$$

对 $\boldsymbol{R}$ 进行全微分,得:

$$\mathrm{d}\boldsymbol{R}=\frac{\partial\boldsymbol{R}}{\partial d}\Delta d+\frac{\partial\boldsymbol{R}}{\partial a}\Delta a+\frac{\partial\boldsymbol{R}}{\partial b}\Delta b+\frac{\partial\boldsymbol{R}}{\partial c}\Delta c \tag{7}$$

则:

$$S_w=\left(\frac{\partial\boldsymbol{R}}{\partial d}\Delta d+\frac{\partial\boldsymbol{R}}{\partial a}\Delta a+\frac{\partial\boldsymbol{R}}{\partial b}\Delta b+\frac{\partial\boldsymbol{R}}{\partial c}\Delta c\right)\boldsymbol{R}^{\mathrm{T}} \tag{8}$$

对 $d^2+a^2+b^2+c^2=1$ 两边微分得:

$$\Delta d=-\frac{a\Delta a+b\Delta b+c\Delta c}{d} \tag{9}$$

将式(9)代入式(8)中,消去 Δd,得:

$$\boldsymbol{S}_w=\begin{bmatrix} 0 & w_3 & -w_2 \\ -w_3 & 0 & w_1 \\ w_2 & -w_1 & 0 \end{bmatrix} \tag{10}$$

式中,

$$w_1=\frac{2}{d}[(d^2+a^2)\Delta a+(ab+dc)\Delta b+(ac-db)\Delta c]$$

$$w_2=\frac{2}{d}[(ab-dc)\Delta a+(d^2+b^2)\Delta b+(da+bc)\Delta c]$$

$$w_3=\frac{2}{d}[(db+ac)\Delta a+(bc-da)\Delta b+(d^2+c^2)\Delta c]$$

令 $\boldsymbol{W}=[w_1\quad w_2\quad w_3]^{\mathrm{T}}$,并整理得:

$$\boldsymbol{W}=\frac{2}{d}\begin{bmatrix} d^2+a^2 & ab+dc & ac-db \\ ab-dc & d^2+b^2 & da+bc \\ db+ac & bc-da & d^2+c^2 \end{bmatrix}\begin{bmatrix}\Delta a\\ \Delta b\\ \Delta c\end{bmatrix} \tag{11}$$

令:

$$\boldsymbol{C}=\frac{2}{d}\begin{bmatrix} d^2+a^2 & ab+dc & ac-db \\ ab-dc & d^2+b^2 & da+bc \\ db+ac & bc-da & d^2+c^2 \end{bmatrix} \tag{12}$$

式中,$\boldsymbol{C}$ 有唯一逆矩阵:

$$\boldsymbol{C}^{-1}=\frac{1}{2}\begin{bmatrix} d & -c & b \\ c & d & -a \\ -b & a & d \end{bmatrix} \tag{13}$$

则式(11)变为:

$$\begin{bmatrix}\Delta a\\ \Delta b\\ \Delta c\end{bmatrix}=\boldsymbol{C}^{-1}\boldsymbol{W}=\frac{1}{2}\begin{bmatrix} dw_1-cw_2+bw_3 \\ cw_1+dw_2-aw_3 \\ -bw_1+aw_2+dw_3 \end{bmatrix} \tag{14}$$

从式(14)中可看到,P-H 法的最大特点是通过求解中间参数 $\boldsymbol{W}$ 反算 d、a、b、c 的改正数,其本质上是一组基于反对称矩阵的求解方案。

1.2　P-H 法相对定向模型

本文通过相对定向模型来确定两幅遥感图像在拍摄时刻的相对姿态。两幅有重叠遥感图像中的某一个点与摄像机在两个拍摄时刻位置形成一个面。利用共面约束条件,可以得到如下方程[19]:

$$F=\begin{vmatrix} B_x & B_y & B_z \\ X_t & Y_t & Z_t \\ p & q & r \end{vmatrix}=0 \tag{15}$$

式中,B_x、B_y、B_z 为摄像机拍摄时的空间位置 B 在 x、y、z 方向上的分量;$(X_t,\ Y_t,\ Z_t)$ 为地面点在 t 时刻的像空间辅助坐标(该坐标系以 t 时刻的像空间坐标系为基准);p、q、r 为同一地面点在 $t+1$ 时刻的像空间辅助坐标。定义:

$$\begin{bmatrix}p\\ q\\ r\end{bmatrix}=\boldsymbol{R}\begin{bmatrix}X_{t+1}\\ Y_{t+1}\\ Z_{t+1}\end{bmatrix} \tag{16}$$

式中,$(X_{t+1},\ Y_{t+1},\ Z_{t+1})$ 为 $t+1$ 时刻的像空间坐标;$\boldsymbol{R}$ 为 $t+1$ 时刻的图像相对 t 时刻的图像的旋转矩阵,通过该矩阵可获得相对姿态参数 φ、κ、ω。

对式(15)进行微分求导得:

$$\mathrm{d}F=\frac{\partial F}{\partial B_y}\mathrm{d}B_y+\frac{\partial F}{\partial B_z}\mathrm{d}B_z+\frac{\partial F}{\partial p}\mathrm{d}p+\frac{\partial F}{\partial q}\mathrm{d}q+\frac{\partial F}{\partial r}\mathrm{d}r \tag{17}$$

式中,$\frac{\partial F}{\partial B_y}=Z_tp-X_tr$;$\frac{\partial F}{\partial B_z}=X_tq-Y_tp$;$\frac{\partial F}{\partial p}=B_yZ_t-B_zY_t$;$\frac{\partial F}{\partial q}=B_zX_t-B_xZ_t$;$\frac{\partial F}{\partial r}=B_xY_t-B_yX_t$。

根据 $\boldsymbol{R}$ 的正交性和 P-H 法的推导结果,对式(16)进行微分求导得:

$$\begin{bmatrix} \mathrm{d}p \\ \mathrm{d}q \\ \mathrm{d}r \end{bmatrix} = \boldsymbol{S}_w \boldsymbol{R} \begin{bmatrix} X_{t+1} \\ Y_{t+1} \\ Z_{t+1} \end{bmatrix} = \begin{bmatrix} qw_3 - rw_2 \\ rw_1 - pw_3 \\ pw_2 - qw_1 \end{bmatrix} \tag{18}$$

将式(15)线性化,并结合式(17)、(18)得:

$$\begin{aligned} F = F_0 + \mathrm{d}F = F_0 + (pZ_t - rX_t)\mathrm{d}B_y + \\ (qX_t - pY_t)\mathrm{d}B_z + (r(B_zX_t - B_xZ_t) - \\ q(B_xY_t - B_yX_t))w_1 + (p(B_xY_t - B_yX_t) - \\ r(B_yZ_t - B_zY_t))w_2 + (q(B_yZ_t - B_zY_t) - \\ p(B_zX_t - B_xZ_t))w_3 \end{aligned} \tag{19}$$

式中,

$$\boldsymbol{A} = \begin{bmatrix} pZ_t - rX_t \\ qX_t - pY_t \\ r(B_zX_t - B_xZ_t) - q(B_xY_t - B_yX_t) \\ p(B_xY_t - B_yX_t) - r(B_yZ_t - B_zY_t) \\ q(B_yZ_t - B_zY_t) - p(B_zX_t - B_xZ_t) \end{bmatrix}^{\mathrm{T}};$$

$\boldsymbol{X} = [\mathrm{d}B_y \quad \mathrm{d}B_z \quad w_1 \quad w_2 \quad w_3]^{\mathrm{T}}$;$\boldsymbol{L} = F_0 - F$。

由于观测值存在误差,式(19)的误差方程应变为:

$$\boldsymbol{V} = \boldsymbol{A}\boldsymbol{X} - \boldsymbol{L} \tag{20}$$

根据最小二乘法平差原理,可列出间接平差的法方程:

$$\boldsymbol{A}^{\mathrm{T}}\boldsymbol{P}\boldsymbol{A}\boldsymbol{X} = \boldsymbol{A}^{\mathrm{T}}\boldsymbol{P}\boldsymbol{L} \tag{21}$$

式中,定义每个像点获取的权重是相同的,即 $\boldsymbol{P}=\boldsymbol{I}$,则未知数的向量解为:

$$\boldsymbol{X} = (\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A})^{-1}\boldsymbol{A}^{\mathrm{T}}\boldsymbol{L} \tag{22}$$

将求得的 w_1、w_2、w_3 代入式(14),求出单位四元数的改正数,得到更新值,再重复式(16)至式(22)的步骤,直至满足迭代条件为止。

2 FPGA 实现

2.1 硬件架构

根据 FPGA 的并行处理特性和最小二乘算法的特点,提出如图 1 所示的 FPGA 硬件架构。图 1 中,实线框代表数值正在计算(结果不可调用),虚线框代表数值计算完成(结果可调用)。不同颜色箭头代表各变量的去向。$T_1 - T_0$ 为一次迭代所需时间,在两次迭代之间($T_1 \sim T_2$),复位信号有效,各模块计算结果清零,并等待下一次迭代开始,每次迭代结果作为下一次迭代的输入,直至迭代结果满足规定阈值。当若干点对输入到系数模块中,采用串行处理,通过牺牲部分处理速度来减少大量的资源消耗,如 p、q、r、$\boldsymbol{A}$、$\boldsymbol{L}$、$\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}$、$\boldsymbol{A}^{\mathrm{T}}\boldsymbol{L}$、$\boldsymbol{X}$。对于计算复杂、时序要求高的模块,则采用并行处理,通过牺牲部分硬件资源来提高处理速度,如 $\boldsymbol{B}^{-1}$。这种策略使处理速度和资源消耗达到平衡。下面详细介绍各个子模块。

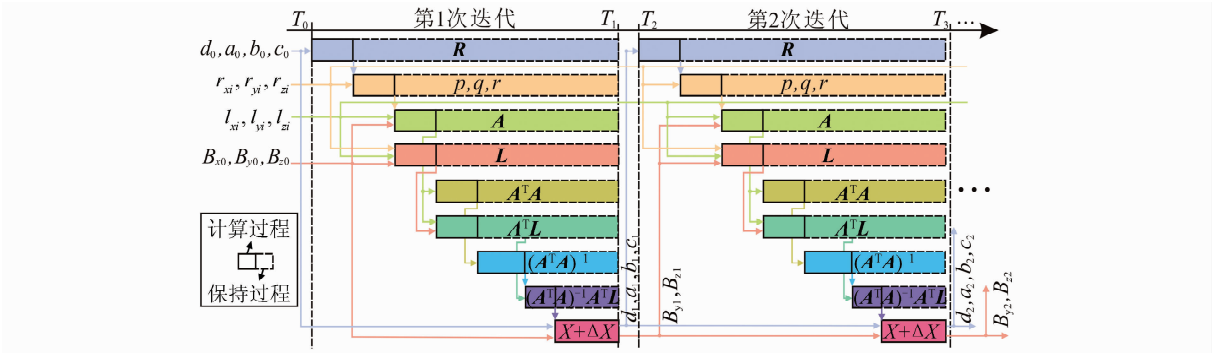


图 1 FPGA 硬件架构

Fig. 1 FPGA Architecture

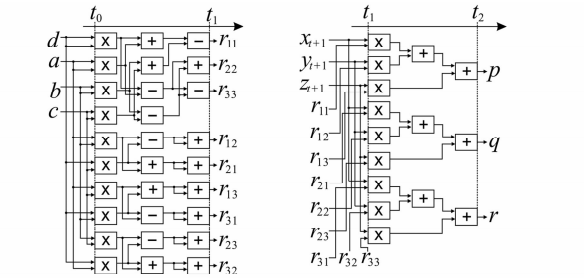
2.2 子模块

2.2.1 系数模块($\boldsymbol{R}$、p、q、r、$\boldsymbol{A}$、$\boldsymbol{L}$、$\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}$、$\boldsymbol{A}^{\mathrm{T}}\boldsymbol{L}$)

FPGA 在计算 $\boldsymbol{R}$ 时(图 2(a)),采用了 3 层结构,第一层为 9 个并行乘法,$T_1 - T_0$ 为乘法运行时间;第二层为 10 个并行加/减法,$T_2 - T_1$ 为加/减法运行时间;第三层为 9 个并行加/减法,运行时间为 $T_3 - T_2$,经过 3 层运算后,直接输出 $\boldsymbol{R}$。计算 p、q、r 时(图 2(b)),同样采用了 3 层结构,第一层为 9 个并行乘法,第二、三层都是 3 个并行加法。在串行计算中,只需调用一次该模块即可。

图 3 为系数 $\boldsymbol{A}$、$\boldsymbol{L}$ 的模块,$\boldsymbol{A}$ 模块有 4 层结构,共有 16 个乘法器,8 个减法器。$\boldsymbol{L}$ 模块共有 5 层结构,其中有 9 个乘法器,5 个加/减法器。$\boldsymbol{L}$ 模块比 $\boldsymbol{A}$ 模块多一层加法运算,因此 $\boldsymbol{L}$ 运行时间比 $\boldsymbol{A}$ 多出一个加法器的时间。

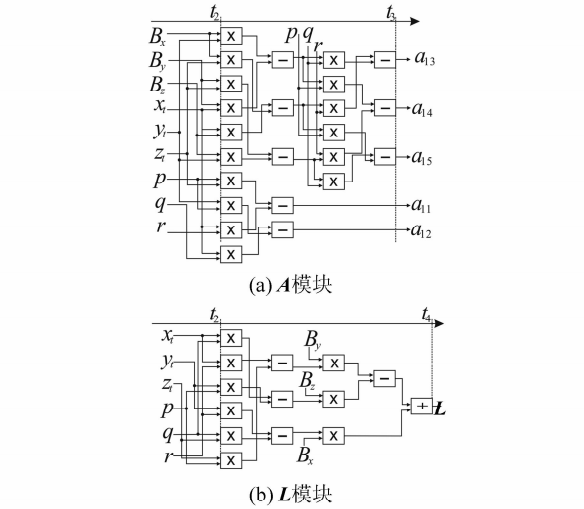
图 4(a)为乘加子模块(MD_i),用于计算结构如"$a_1b_1 + a_2b_2 + \cdots + a_ib_i$"的结果。由于该架构使用 9 组点对($i=9$),在 $\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}$ 和 $\boldsymbol{A}^{\mathrm{T}}\boldsymbol{L}$ 的计算中,均基于 MD_9 子模块运算。在串行执行中,$\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}$ 使用了 5 个该模块,$\boldsymbol{A}^{\mathrm{T}}\boldsymbol{L}$ 则使用了 1 个。



(a) 计算$\boldsymbol{R}$　　(b) 计算p、q、r

图 2　$\boldsymbol{R}$、p、q、r 子模块

Fig. 2　$\boldsymbol{R}$ and p, q, r Sub-modules

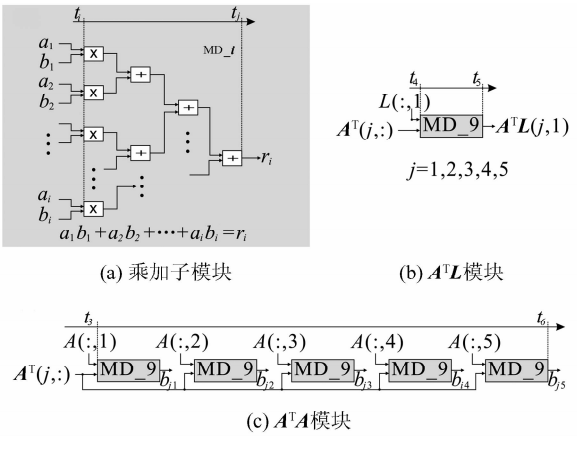


(a) $\boldsymbol{A}$模块

(b) $\boldsymbol{L}$模块

图 3　$\boldsymbol{A}$、$\boldsymbol{L}$ 子模块

Fig. 3　$\boldsymbol{A}$ and $\boldsymbol{L}$ Sub-modules



(a) 乘加子模块　　(b) $\boldsymbol{A}^{\mathrm{T}}\boldsymbol{L}$模块

(c) $\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}$模块

图 4　$\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}$、$\boldsymbol{A}^{\mathrm{T}}\boldsymbol{L}$ 子模块

Fig. 4　$\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}$ and $\boldsymbol{A}^{\mathrm{T}}\boldsymbol{L}$ Sub-modules

2.2.2　矩阵求逆模块($\boldsymbol{B}^{-1}=(\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A})^{-1}$)

在 FPGA 的矩阵求逆中，若直接通过伴随矩阵方式，会消耗大量硬件资源，增加计算时间，不利于实时性要求高的场景，因此，需要从另外角度进行矩阵求逆。一般的矩阵求逆方法有 LU 分解、LDLT(lower-diagonally-lower-transpose)分解及乔里斯基分解等。这些方法虽然能减少计算量，但计算速度有待提高，并行度不高。针对 FPGA 并行处理的特性，提出 LU 分解-分块算法[20]的矩阵求逆。由于 $\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}$ 的大小为 5×5，首先定义：

$$\boldsymbol{B}=\begin{bmatrix}b_{11}&b_{12}&b_{13}&b_{14}&b_{15}\\b_{21}&b_{22}&b_{23}&b_{24}&b_{25}\\b_{31}&b_{32}&b_{33}&b_{34}&b_{35}\\b_{41}&b_{42}&b_{43}&b_{44}&b_{45}\\b_{51}&b_{52}&b_{53}&b_{54}&b_{55}\end{bmatrix}=\begin{bmatrix}\boldsymbol{B}_{11}&\boldsymbol{B}_{12}\\\boldsymbol{B}_{21}&\boldsymbol{B}_{22}\end{bmatrix}\tag{23}$$

式中，$\boldsymbol{B}_{11}$ 大小为 3×3；$\boldsymbol{B}_{12}$ 为 3×2；$\boldsymbol{B}_{21}$ 为 2×3；$\boldsymbol{B}_{22}$ 为 2×2。

LU 分解-分块算法为：

$$\begin{bmatrix}\boldsymbol{B}_{11}&\boldsymbol{B}_{12}\\\boldsymbol{B}_{21}&\boldsymbol{B}_{22}\end{bmatrix}=\begin{bmatrix}\boldsymbol{L}_{11}&\boldsymbol{0}\\\boldsymbol{L}_{21}&\boldsymbol{L}_{22}\end{bmatrix}\begin{bmatrix}\boldsymbol{U}_{11}&\boldsymbol{U}_{12}\\\boldsymbol{0}&\boldsymbol{U}_{22}\end{bmatrix}\tag{24}$$

式中，$\boldsymbol{B}_{11}=\boldsymbol{L}_{11}\boldsymbol{U}_{11}$；$\boldsymbol{B}_{12}=\boldsymbol{L}_{11}\boldsymbol{U}_{12}$；$\boldsymbol{B}_{21}=\boldsymbol{L}_{21}\boldsymbol{U}_{11}$；$\boldsymbol{B}_{22}=\boldsymbol{L}_{21}\boldsymbol{U}_{12}+\boldsymbol{L}_{22}\boldsymbol{U}_{22}$。

由式(24)计算得：

$$\boldsymbol{L}_{11}=\begin{bmatrix}1&0&0\\l_{21}&1&0\\l_{31}&l_{32}&1\end{bmatrix},\boldsymbol{U}_{11}=\begin{bmatrix}u_{11}&u_{12}&u_{13}\\0&u_{22}&u_{23}\\0&0&u_{33}\end{bmatrix}\tag{25}$$

式中 $u_{11}=b_{11}$；$u_{12}=b_{12}$；$u_{13}=b_{13}$；$l_{21}=b_{21}/b_{11}$；$l_{31}=b_{31}/b_{11}$；$u_{22}=b_{22}-l_{21}u_{12}$；$u_{23}=b_{23}-l_{21}u_{13}$；$l_{32}=(b_{32}-l_{31}u_{12})/u_{22}$；$u_{33}=b_{33}-l_{31}u_{13}-l_{32}u_{23}$。

计算 $\boldsymbol{L}_{11}$ 和 $\boldsymbol{U}_{11}$ 后，可以得到：

$$\boldsymbol{U}_{12}=\boldsymbol{L}_{11}^{-1}\boldsymbol{B}_{12},\boldsymbol{L}_{21}=\boldsymbol{B}_{21}\boldsymbol{U}_{11}^{-1}\tag{26}$$

令：

$$\boldsymbol{B}'_{22}=\boldsymbol{B}_{22}-\boldsymbol{L}_{21}\boldsymbol{U}_{12}=\begin{bmatrix}b'_{11}&b'_{12}\\b'_{21}&b'_{22}\end{bmatrix}\tag{27}$$

则：

$$\boldsymbol{L}_{22}=\begin{bmatrix}1&0\\l'_{21}&1\end{bmatrix},\boldsymbol{U}_{22}=\begin{bmatrix}u'_{11}&u'_{12}\\0&u'_{22}\end{bmatrix}\tag{28}$$

式中，$u'_{11}=b'_{11}$；$u'_{12}=b'_{12}$；$l'_{21}=b'_{21}/u'_{11}$；$u'_{22}=b'_{22}-l'_{21}u'_{12}$。

因此，有：

$$\boldsymbol{B}^{-1}=\begin{bmatrix}\boldsymbol{U}_{11}&\boldsymbol{U}_{12}\\\boldsymbol{0}&\boldsymbol{U}_{22}\end{bmatrix}^{-1}\begin{bmatrix}\boldsymbol{L}_{11}&\boldsymbol{0}\\\boldsymbol{L}_{21}&\boldsymbol{L}_{22}\end{bmatrix}^{-1}\tag{29}$$

式中，$\begin{bmatrix}\boldsymbol{U}_{11}&\boldsymbol{U}_{12}\\\boldsymbol{0}&\boldsymbol{U}_{22}\end{bmatrix}^{-1}=\begin{bmatrix}\boldsymbol{U}_{11}^{-1}&\boldsymbol{M}_{12}^{-1}\\\boldsymbol{0}&\boldsymbol{U}_{22}^{-1}\end{bmatrix}$，

$\boldsymbol{M}_{12}^{-1}=-\boldsymbol{U}_{11}^{-1}\boldsymbol{U}_{12}\boldsymbol{U}_{22}^{-1}$；

$\begin{bmatrix}\boldsymbol{L}_{11}&\boldsymbol{0}\\\boldsymbol{L}_{21}&\boldsymbol{L}_{22}\end{bmatrix}^{-1}=\begin{bmatrix}\boldsymbol{L}_{11}^{-1}&\boldsymbol{0}\\\boldsymbol{N}_{21}^{-1}&\boldsymbol{L}_{22}^{-1}\end{bmatrix}$，

$\boldsymbol{N}_{21}^{-1}=-\boldsymbol{L}_{22}^{-1}\boldsymbol{L}_{21}\boldsymbol{L}_{11}^{-1}$。

根据式(23)到式(29),LU分解-分块算法的流程见图5。

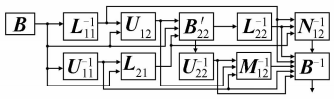
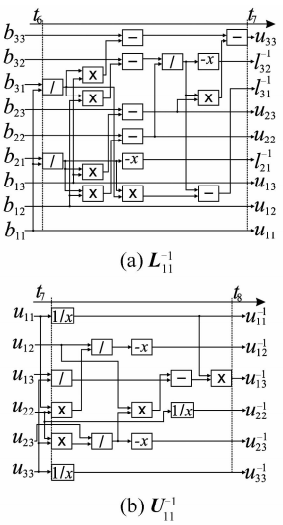


图5 LU分解-分块算法流程图

Fig. 5 Flowchart of LU Decomposition Block Algorithm

下面介绍各变量在FPGA中的实现。图6为$\boldsymbol{L}_{11}^{-1}$和$\boldsymbol{U}_{11}^{-1}$的FPGA实现过程,$\boldsymbol{B}_{11}$为输入数据,由于$\boldsymbol{L}_{11}^{-1}$和$\boldsymbol{U}_{11}^{-1}$分别属于上三角型矩阵和下三角型矩阵,因此只需求部分元素,其余元素为常量。图中"$-x$"为符号位取反模块,执行不需要时间。



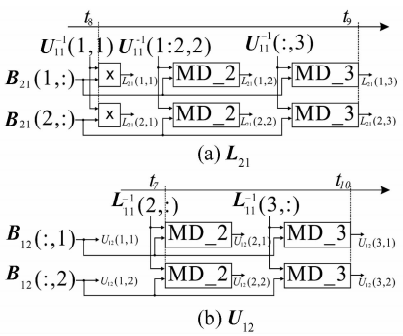
(a) $\boldsymbol{L}_{11}^{-1}$

(b) $\boldsymbol{U}_{11}^{-1}$

图6 $\boldsymbol{L}_{11}^{-1}$和$\boldsymbol{U}_{11}^{-1}$子模块

Fig. 6 $\boldsymbol{L}_{11}^{-1}$ and $\boldsymbol{U}_{11}^{-1}$ Sub-modules

图7为式(27)的FPGA实现过程,其中分别使用了4个MD_2和MD_3模块。



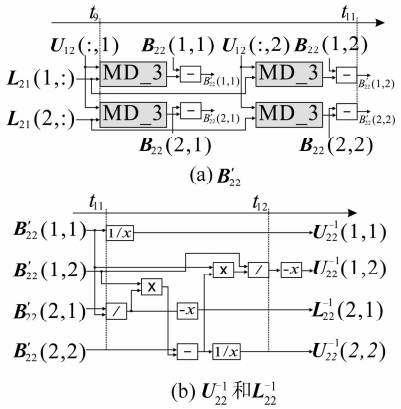
(a) $\boldsymbol{L}_{21}$

(b) $\boldsymbol{U}_{12}$

图7 $\boldsymbol{L}_{21}$和$\boldsymbol{U}_{12}$子模块

Fig. 7 $\boldsymbol{L}_{21}$ and $\boldsymbol{U}_{12}$ Sub-modules

图8为$\boldsymbol{L}_{22}^{-1}$和$\boldsymbol{U}_{22}^{-1}$的FPGA实现过程,其中$\boldsymbol{L}_{21}$、$\boldsymbol{U}_{12}$和$\boldsymbol{B}_{22}$为输入,直接输出$\boldsymbol{L}_{22}^{-1}$和$\boldsymbol{U}_{22}^{-1}$中的元素。



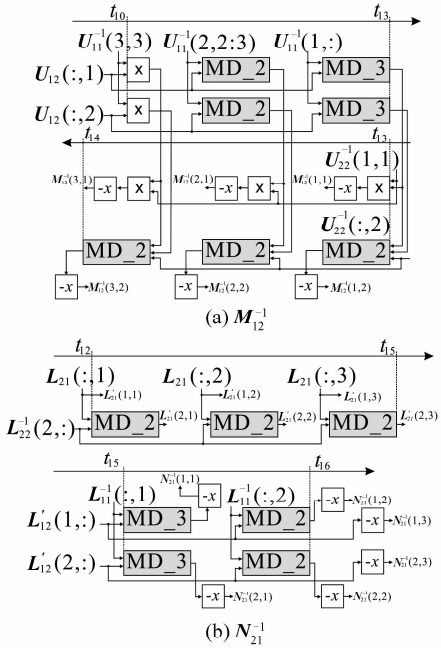
(a) $\boldsymbol{B}'_{22}$

(b) $\boldsymbol{U}_{22}^{-1}$和$\boldsymbol{L}_{22}^{-1}$

图8 $\boldsymbol{L}_{22}^{-1}$和$\boldsymbol{U}_{22}^{-1}$子模块

Fig. 8 $\boldsymbol{L}_{22}^{-1}$ and $\boldsymbol{U}_{22}^{-1}$ Sub-modules

图9为式(30)中$\boldsymbol{M}_{12}^{-1}$和$\boldsymbol{N}_{21}^{-1}$的FPGA实现过程,其中输入分别为$\boldsymbol{U}_{11}^{-1}$、$\boldsymbol{U}_{12}$、$\boldsymbol{U}_{22}^{-1}$和$\boldsymbol{L}_{22}^{-1}$、$\boldsymbol{L}_{21}$、$\boldsymbol{L}_{11}^{-1}$。



(a) $\boldsymbol{M}_{12}^{-1}$

(b) $\boldsymbol{N}_{21}^{-1}$

图9 $\boldsymbol{M}_{12}^{-1}$和$\boldsymbol{N}_{21}^{-1}$子模块

Fig. 9 $\boldsymbol{M}_{12}^{-1}$ and $\boldsymbol{N}_{21}^{-1}$ Sub-modules

图10为$\boldsymbol{U}_{11}^{-1}\boldsymbol{L}_{11}^{-1}$和$\boldsymbol{M}_{12}^{-1}\boldsymbol{N}_{21}^{-1}$的FPGA实现过程,使用了多个并行MD_3和MD_2模块,属于计算$\boldsymbol{B}^{-1}$的中间结果。

图11为FPGA实现$\boldsymbol{B}^{-1}$的最后一步,图11(a)、11(d)分别为$\boldsymbol{B}^{-1}$(1∶3,1∶3)、$\boldsymbol{B}^{-1}$(1∶3,4∶5)、$\boldsymbol{B}^{-1}$(4∶5,1∶3)、$\boldsymbol{B}^{-1}$(4∶5,4∶5)。图6至图11为求解$\boldsymbol{B}^{-1}$的全部过程。

2.2.3 改正数模块

图12(a)为式(9)和式(14)的实现过程;图12(b)为式(22)中$\boldsymbol{X}$的实现过程;图12(c)为更新后的a、b、c、d、B_y、B_z。重复图2至图12过程,直至Δa、Δb、Δc、Δd满足给定阈值。

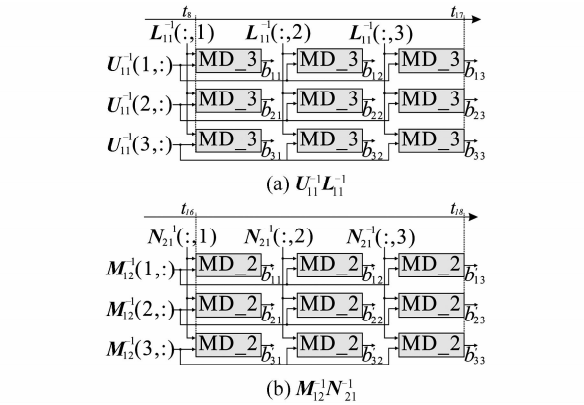


图 10　$U_{11}^{-1}L_{11}^{-1}$ 和 $M_{12}^{-1}N_{21}^{-1}$ 子模块

Fig. 10　$U_{11}^{-1}L_{11}^{-1}$ and $M_{12}^{-1}N_{21}^{-1}$ Sub-modules

2.3　Modelsim 仿真

使用 Verilog HDL 硬件语言在 VIVADO 软件实现上述各个模块,经过综合后,即可进行 Modelsim 仿真。图 13 为顶层模块的仿真波形,其中 L_X_t、L_Y_t、L_Z_t 和 R_X_t、R_Y_t、R_Z_t 为输入的点对,d_3、a_3、b_3、c_3 为每次迭代后的更新结果。经过 7 次迭代后,it0_en 信号拉低,表示迭代终止。图 14 为 LU 分解-分块算法的仿真波形,每次迭代都需要进行矩阵求逆,图 14 中显示的数值是第 7 次的求逆结果。

3　对比分析

FPGA 参数为 Xilinx, XC7VX1140T, 其主要资源有触发器(FF)1 424 000 个,查找表(LUT)712 000 个,块内存为 1 880 kB,3 360 个 DSP 单元等。

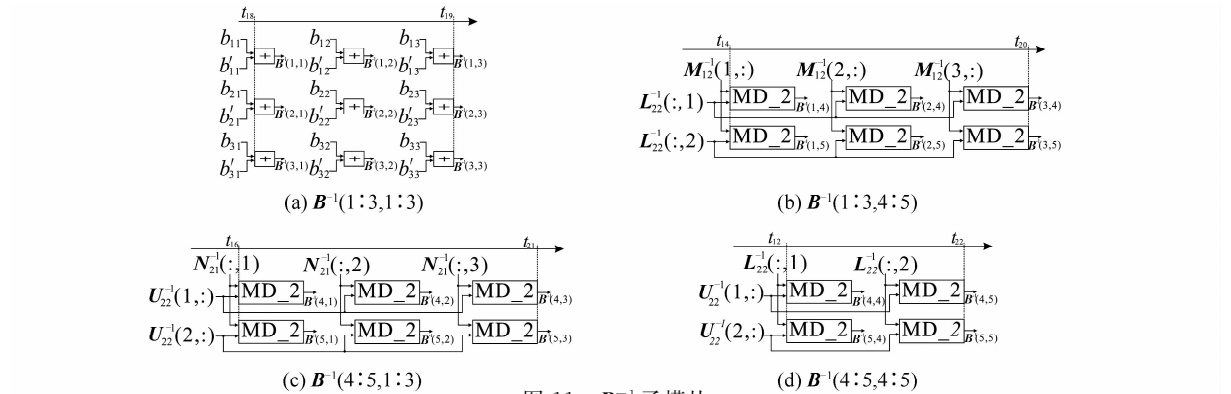


图 11　B^{-1} 子模块

Fig. 11　B^{-1} Sub-module

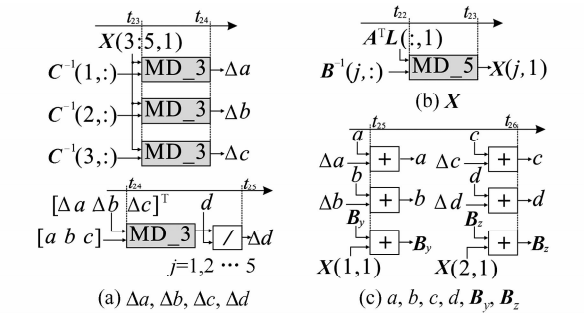


图 12　改正数模块

Fig. 12　Correction Module

PC 机为 Win10 系统,CPU 为 Intel (R) Core (TM) i7-4770 CPU@3.40 GHz,内存为 8 GB,软件为 Dev-C^{++} 5.11。

FPGA 端和 PC 端输入参数为两幅图像的 9 个点对(表 1)。迭代终止条件为 w_1、w_2、w_3 都大于或等于 1.0×10^{-7},迭代结果见表 2。由表 2 可知,PC1 与 PC2 的最大偏差在 ω,约为 2.4×10^{-5};PC2 与 FPGA 的最大偏差值约为 5.0×10^{-14}。基于 P-H 法的迭代次数比欧拉角的少 13

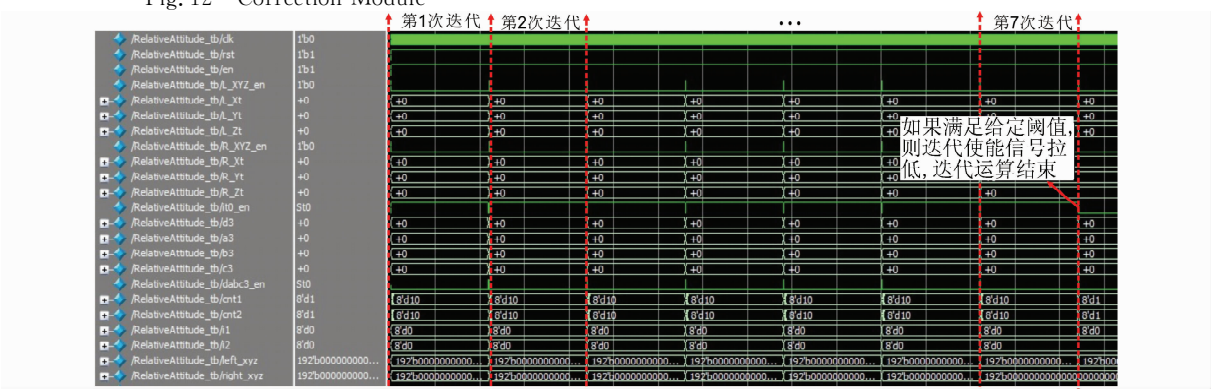


图 13　迭代过程的仿真波形

Fig. 13　Simulation Waveform of Iteration Processing

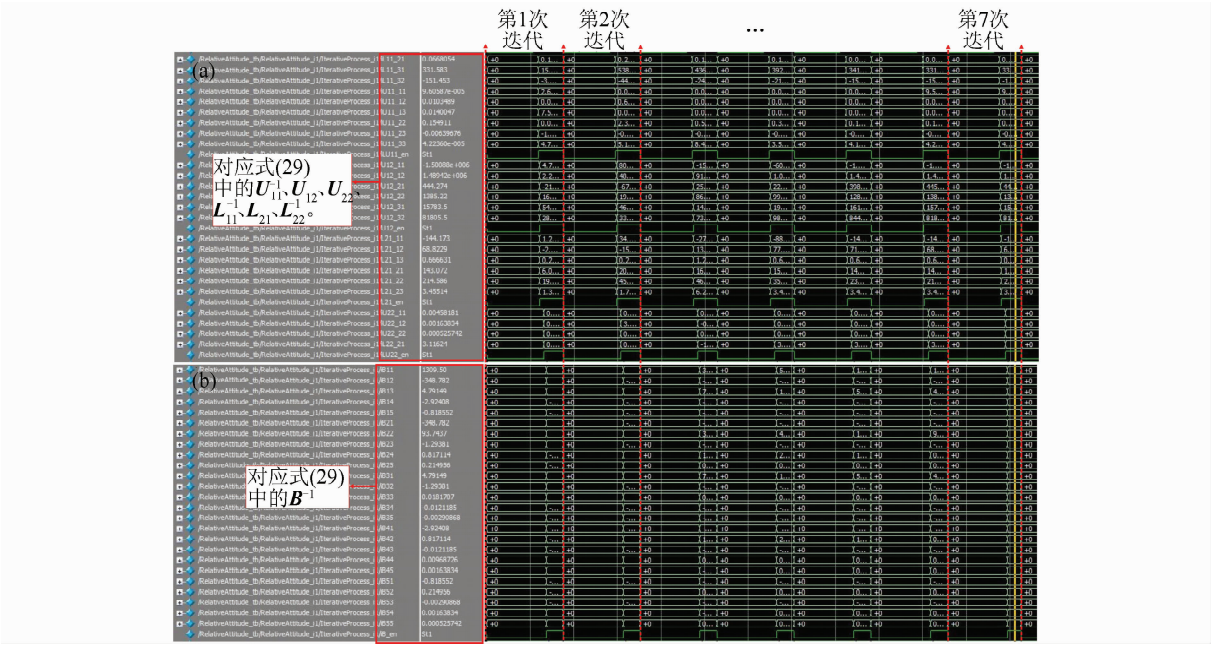


图 14 LU 分解-分块算法的仿真波形

Fig. 14 Simulation Waveform of LU Decomposition Block Algorithm

表 1 两幅图像点对(f=0.1 m)

Tab. 1 Point Correspondences of Two Images (f=0.1 m)

点号	左图像		右图像	
	行号	列号	行号	列号
1	−37.104	−20.849	−43.682	−21.725
2	−34.718	−21.669	−41.243	−22.652
3	−32.359	−22.520	−38.839	−23.610
4	−29.959	−23.372	−36.396	−24.567
5	−27.631	−24.187	−34.030	−25.483
6	−36.121	−18.675	−42.539	−19.529
7	−33.601	−19.385	−39.958	−20.350
8	−31.343	−20.141	−37.657	−21.206
9	−28.945	−20.954	−35.218	−22.125

次。在计算速度方面,PC1 与 PC2 的运行时间分别为 2.269 ms 和 2.873 ms,FPGA 的计算运行时间为0.308 ms(100 MHz),加速比约为 10 倍。FPGA 资源消耗情况见表 3。表 3 中资源消耗最多的是 DSP48,约为 80.7%。由于整个模型属于数值运算,调用了大量的浮点运算 IP 核。其次是 LUT 和 FF,百分比约为 50%。所选的 FPGA(V7 xc7vx1140tflg1930-1)满足整个模型所需的硬件资源。

4 结语与展望

本文针对遥感图像复杂算法星上实时处理技术难点,提出了基于 FPGA 的 P-H 法星上相对姿态解算。对比试验发现,在算法层面,旋转矩阵的计算由基于欧拉角转换为基于单位四元数,采用 P-H 法平差模型转换为间接平差。在迭代处理中,直接取d=1和a=b=c=0,避免了初值估算

表 2 结果对比

Tab. 2 Results Comparison

参数	PC1(欧拉角)	PC2(P-H 法)	FPGA(P-H 法)	|PC1−PC2|	|PC2−FPGA|
φ(picth) 航向倾角	0.043 373 3 6 510 974	0.043 373 3 6 316 752	0.043 373 3 6 316 747	1.9×10^{-9}	5×10^{-14}
κ(yaw) 旁向倾角	0.044 063 6 4 319 247	0.044 063 6 5 064 706	0.044 063 6 5 064 708	7.5×10^{-9}	2×10^{-14}
ω(roll) 像片旋角	0.021 523 4 0 090 194	0.021 547 6 7 515 801	0.021 547 6 7 515 795	2.4×10^{-5}	5×10^{-14}
迭代次数	20	7	7	13	0
运行时间/ms	2.269 (3.40 GHz)	2.873 (3.40 GHz)	0.308 (100 MHz)	/	/

表 3　FPGA 硬件资源消耗

Tab. 3　Utilization of FPGA Resources

资源	消耗情况	占总资源百分比/%
FF	656 464	46.1
LUT	382 344	53.7
Memory LUT	45 596	16.1
DSP48	2 712	80.7

和大量的三角函数计算，使迭代次数减少了 13 次；在矩阵求逆方面，提出 LU 分解-分块算法。该算法不仅使最小单元的矩阵维度由 5×5 降到了 3×3，更提高了算法的并行性和处理速度；在 FPGA 方面，通过采用 64 位的双精度浮点数据结构，保证了计算结果的精度。如本方法在 PC 与 FPGA 计算结果偏差仅为 5.0×10^{-14}。另外，采用串行、并行相结合的策略，在系数（$p,q,r,\boldsymbol{A},\boldsymbol{L},\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A},\boldsymbol{A}^{\mathrm{T}}\boldsymbol{L}$）模块和改正数（$\boldsymbol{X}$）模块中采用的是串行方式处理，在矩阵求逆模块和更新模块则采用并行处理，这种策略使得处理速度和资源消耗达到平衡。

参　考　文　献

[1] Zhou Guoqing, Baysal O, Kaye J. Concept Design of Future Intelligent Earth Observing Satellites[J]. *International Journal of Remote Sensing*, 2004, 25(14): 2 667-2 685

[2] Li Deren, Shen Xin. On Intelligent Earth Observation Systems[J]. *Science of Surveying and Mapping*, 2005, 30(4): 9-11 (李德仁, 沈欣. 论智能化对地观测系统[J]. 测绘科学, 2005, 30(4): 9-11)

[3] Zhang Bing. Intelligent Remote Sensing Satellite System[J]. *Journal of Remote Sensing*, 2011, 15(3): 415-431 (张兵. 智能遥感卫星系统[J]. 遥感学报, 2011, 15(3): 415-431)

[4] Li Deren, Wang Mi, Shen Xin, et al. From Earth Observation Satellite to Earth Observation Brain[J]. *Geomatics and Information Science of Wuhan University*, 2017, 42(2): 143-149 (李德仁, 王密, 沈欣, 等. 从对地观测卫星到对地观测脑[J]. 武汉大学学报·信息科学版, 2017, 42(2): 143-149)

[5] Li Deren, Gong Jianya, Shao Zhenfeng. From Digital Earth to Smart Earth[J]. *Geomatics and Information Science of Wuhan University*, 2010, 35(2): 127-132 (李德仁, 龚健雅, 邵振峰. 从数字地球到智慧地球[J]. 武汉大学学报·信息科学版, 2010, 35(2): 127-132)

[6] Du Liebo, Xiao Xuemin, Lu Qin, et al. An Implementation Scheme for JPEG2000 Satellite-Borne Remote Sensing Image Compression Based on FPGA + Multi-DSPs[J]. *Journal of Test and Measurement Technology*, 2008, 22(6): 478-482 (杜列波, 肖学敏, 鲁琴, 等. 基于 FPGA＋多 DSP 的 JPEG2000 星载遥感图像压缩实现方案[J]. 测试技术学报, 2008, 22(6): 478-482)

[7] Gong Jianya, Zhong Yanfei. Survey of Intelligent Optical Remote Sensing Image Processing[J]. *Journal of Remote Sensing*, 2016, 20(5): 733-747 (龚健雅, 钟燕飞. 光学遥感影像智能化处理研究进展[J]. 遥感学报, 2016, 20(5): 733-747)

[8] Gao Lining, Long Teng. On-Board Spaceborne Real-Time Digital Signal Processing System[C]. Computer Engineering and Applications, Nanjing, China, 2008 (高立宁, 龙腾. 基于 FPGA 的星上实时信息处理系统[C]. 全国信号处理与应用学术会议, 南京, 2008)

[9] You Zheng, Dai Mi. "Hangtian Tsinghua_1" Micro-Satellite and Its Image Processing[J]. *Journal of Remote Sensing*, 2001, 5(3): 177-182 (尤政, 戴汨. "航天清华一号"微小卫星及其图像处理[J]. 遥感学报, 2001, 5(3): 177-182)

[10] Williams J A, Dawood A S, Visser S J. FPGA-Based Cloud Detection for Real-Time Onboard Remote Sensing[C]. IEEE International Conference on Field-Programmable Technology, Hong Kong, China, 2002

[11] Gao Kun, Liu Yinghui, Ni Guoqiang, et al. Study on On-Board Real-Time Image Processing Technology of Optical Remote Sensing[J]. *Spacecraft Recovery & Remote Sensing*, 2008, 29(1): 50-54 (高昆, 刘迎辉, 倪国强, 等. 光学遥感图像星上实时处理技术的研究[J]. 航天返回与遥感, 2008, 29(1): 50-54)

[12] Huang Jingjin, Zhou Guoqing. On-Board Detection and Matching of Feature Points[J]. *Remote Sensing*, 2017, 9: 601

[13] Huang Jingjin, Zhou Guoqing, Zhang Dianjun, et al. An FPGA-Based Implementation of Corner Detection and Matching with Outlier Rejection[J]. *International Journal of Remote Sensing*, 2018, doi: 10.1080101431161.2018.1500728

[14] Zhou Guoqing, Jiang Linjun, Huang Jingjin, et al. FPGA-Based On-Board Geometric Calibration for Linear CCD Array Sensors[J]. *Sensors*, 2018, 18(6): 1 794

[15] Yan Li, Nie Qian, Zhao Zhan. Space Resection of Line Scanner CCD Image Based on the Description of Quaternions[J]. *Geomatics and Information Science of Wuhan University*, 2010, 35(2): 201-204 (闫利, 聂倩, 赵展. 利用四元数描述线阵 CCD 影像的空间后方交会[J]. 武汉大学学报·信息科学版, 2010, 35(2): 201-204)

[16] Zhou Yongjun, Deng Caihua. A New Method for

Relative Orientation with Hybrid Genetic Algorithm and Unit Quaternion[J]. *Geomatics and Information Science of Wuhan University*, 2011, 36(6): 670-673 (周拥军, 邓才华. 利用HGA和单位四元数的相对定向解法[J]. 武汉大学学报·信息科学版, 2011, 36(6):670-673)

[17] Ben Jin, Tong Xiaochong, Lv Haiqing. Space Resection Based on the Pope-Hinsken Algorithm[J]. *Journal of Geomatics Science and Technology*, 2011, 28(1): 37-41 (贲进, 童晓冲, 闾海庆. 基于Pope-Hinsken算法的空间后方交会[J]. 测绘科学技术学报, 2011, 28(1):37-41)

[18] Jiang Gangwu, Jiang Ting, Wang Yong, et al. Space Resection Independent of Initial Value Based on Unit Quaternions[J]. *Acta Geodaetica et Cartographica Sinica*, 2007, 36(2):169-175 (江刚武, 姜挺, 王勇, 等. 基于单位四元数的无初值依赖空间后方交会[J]. 测绘学报, 2007, 36(2):169-175)

[19] Wang Zhizhuo. Photogrammetry Principle[M]. Wuhan: Wuhan University Press, 2007(王之卓. 摄影测量原理[M]. 武汉:武汉大学出版社, 2007)

[20] Wu Guiming. Parallel Algorithms and Architectures for Matrix Computations on FPGA[D]. Changsha: National University of Defense Technology, 2011 (邬贵明. FPGA矩阵计算并行算法与结构[D]. 长沙:国防科学技术大学, 2011)

An FPGA-Based P-H Method On-Board Solution for Satellite Relative Attitude

ZHOU Guoqing[1,2] *HUANG Jingjin*[1] *SHU Lei*[1]

1 School of Precision Instrument & Opto-Electronics Engineering, Tianjin University, Tianjin 300072, China

2 Guangxi Key Laboratory for Spatial Information and Geomatics, Guilin University of Technology, Guilin 541004, China

Abstract: Aimed at the situation that the low-level algorithms were implemented in satellite real time processing system for remote sensing image, this paper proposes an FPGA (field programmable gate array) -based P-H method for satellite relative attitude on-board solution. The proposed algorithm not only avoids computations of trigonometric function and estimation of initial value, but also reduces the number of iterations when comparing with the Eulerian angle-based algorithm. The Xilinx FPGA (V7 xc7vx1140tflg1930-1) is selected as the hardware platform for the real-time processing. In FPGA implementation: ①We adopt a 64-bit floating point data structure and a strategy of combination of serial and parallel processing; ②a lower-upper (LU) decomposition-block algorithm is adopted for matrix inversion. The experimental results indicate that the number of iterations of the proposed algorithm is 13 less than the Eulerian angle-based algorithm. The difference of FPGA and PC implementation is about 5.0×10^{-14} and the speedup is about 10, which meets the requirements of precision and speed for on-board image real time processing. The proposed algorithm can be suitable for high real-time image processing field.

Key words: P-H method; unit quaternion; relative attitude; LU decomposition-block algorithm; FPGA

First author: ZHOU Guoqing, PhD, professor, specializes in the theories and methods of intelligent satellite observation systems and LiDAR technology. E-mail: gzhou@glut.edu.cn

Corresponding author: HUANG Jingjin, PhD candidate. E-mail: jingjin_huang@tju.edu.cn

Foundation support: The National Natural Science Foundation of China, No. 41431179; the National Key Research and Development Program of China, No. 2016YFB0502501; Guangxi Innovative Development Grand, No. GuikeAA18118038.