Task-Driven Onboard Real-Time Panchromatic Multispectral Fusion Processing Approach for High-Resolution Optical Remote Sensing Satellite

Zhiqi Zhang, Lu Wei, Shao Xiang, Guangqi Xie, Chuang Liu, and Mingyuan Xu

***Abstract*—Onboard real-time processing of remote sensing satellites is an important means of rapidly obtaining information, and the fusion processing of panchromatic and multispectral data is of great significance for optical satellites. In order to ensure the effect, most traditional algorithms perform statistical analysis or transformation on the entire image first and then perform subsequent processing. There are problems such as high algorithm complexity and resource occupation, and it is difficult to apply to onboard scenarios where the volume and power consumption are strictly limited. Aiming at the requirements of onboard fusion, a real-time processing approach for high-resolution optical satellites is proposed. First, through the implementation of real-time geometric positioning, ROI extraction is completed while the camera is imaging, avoiding the disadvantages of traditional methods for processing large amounts of data; then, based on the principle of object-space consistency, by fine-tuning virtual sensor parameters, the registration of a panchromatic multispectral image is completed in the sensor correction step, so that the relative accuracy of the two can meet the fusion requirements, and time-consuming pixel-level registration processing is avoided; finally, according to the characteristics of the algorithms and embedded hardware, an efficient algorithm mapping strategy is formulated, and deep optimization is implemented to achieve a significant improvement in performance. Experiments show that the performance of this method is improved by 156.23 times compared with the traditional method. Moreover, after building a parallel pipeline, it can meet the real-time fusion processing requirement of completing a 5000 $\times$ 5000 pixels ROI area every 2.4 s.**

***Index Terms*—Onboard, optical remote sensing, panchromatic multispectral fusion, real-time, task-driven.**

Manuscript received 29 June 2023; revised 20 July 2023; accepted 10 August 2023. Date of publication 15 August 2023; date of current version 24 August 2023. This work was supported in part by the National Key R&D Program of China under Grant 2022YFB3902800, in part by the National Natural Science Foundation of China under Grant 61901307, and in part by the Scientific Research Foundation for Doctoral Program of the Hubei University of Technology under Grant BSQD2020054 and Grant XJ2022005901. *(Corresponding author: Shao Xiang.)*

Zhiqi Zhang is with the School of Computer Science, Hubei University of Technology, Wuhan 430068, China, and also with the State Key Laboratory of Information Engineering in Surveying, Mapping, and Remote Sensing, Wuhan University, Wuhan 430079, China (e-mail: zzq540@hbut.edu.cn).

Lu Wei is with the School of Information Science and Engineering, Wuchang Shouyi University, Wuhan 430064, China (e-mail: weilu@wsyu.edu.cn).

Shao Xiang is with the State Key Laboratory of Information Engineering in Surveying, Mapping, and Remote Sensing, Wuhan University, Wuhan 430079, China (e-mail: xiangshao@whu.edu.cn).

Guangqi Xie, Chuang Liu, and Mingyuan Xu are with the School of Computer Science, Hubei University of Technology, Wuhan 430068, China (e-mail: xieguangqi@hbut.edu.cn; liuchuang@hbut.edu.cn; 2010300501@hbut.edu.cn).

Digital Object Identifier 10.1109/JSTARS.2023.3305231

I. Introduction

With the advancement of science and technology, satellite remote sensing technology has also achieved leapfrog development. In recent years, high-resolution optical satellites represented by GeoEye, WorldView, Pleiades, ZY series, GF series, and YG series have been successively launched. It has entered a new era of diversified data acquisition methods and massive information [1], [2]. In order to improve the application efficiency of optical remote sensing satellites, high resolution, large width, and multispectral bands have become the main goals for the development of optical remote sensing satellites in the future. The combination of the above factors leads to a geometric progression in the amount of data acquired by optical remote sensing satellites. While the massive data acquisition capability provides a rich source for subsequent applications and services, it also brings increasing pressure on satellite–ground data transmission links and ground systems.

In the time-sensitive application field of high-resolution remote sensing data, the proportion of received remote sensing data that can be processed in time and effectively used by users is relatively low, and there is a problem that the concerned information is hidden by massive data [3]. The traditional optical satellite remote sensing data acquisition and processing procedure includes three main steps: "data acquisition-satellite data transmission-ground reception and processing." In the process of processing, distribution, and application, the focus is on the data itself rather than how to apply it efficiently. In this process, multiple steps such as compression, decompression, communication, reception, recording, formatting, processing, and storage are required, which consume a large amount of transmission, computing, and storage resources. The delay in obtaining information often exceeds tens of minutes, and the timeliness is low. At the same time, users' expectations for time-sensitive applications such as area monitoring, target positioning, target tracking, disaster response, and emergency rescue continue to increase. In order to make full use of the limited satellite–ground transmission bandwidth and satellite transit time window and shorten the information acquisition delay, it is urgently necessary to carry a computing platform on the satellite and deploy real-time processing on it and migrate key algorithms to the satellite to realize real-time processing and Information extraction [4], [5], thereby effectively reducing the amount of data and reducing the pressure of satellite–ground data transmission and ground processing [6]. With this goal, scholars have carried out a series of research on efficient remote sensing algorithms that have the potential to be applied on board [7], [8], [9], [10], [11], [12], [13].

However, the particularity of the space environment greatly limits the onboard computing platform. For remote sensing satellites in low-Earth orbit, the temperature difference between the sunny side and the shady side can reach more than 300°, and the extreme temperature exceeds the normal working temperature range of typical electronic equipment. It is necessary to use auxiliary heating and heat dissipation methods to make electronic equipment work normally. In addition, there are high-energy ionizing radiations such as galactic cosmic rays and solar cosmic rays in the space environment, which are collectively referred to as space radiation [14]. The impact of space radiation on the onboard processing platform mainly includes the total dose effect [15], [16], [17] and single event effect [18], which are the main reasons for the failure of space equipment [19]. Accordingly, it is necessary to strengthen and protect electronic equipment [20], [21], [22], [23]. Various reinforcement and protection methods need to occupy a certain volume and weight. Commonly used redundant backup means will also bring additional power consumption. Overall, it further limits the size, power consumption, and performance of onboard computing devices. Therefore, onboard processing should be task-driven, utilizing limited onboard computing resources to process closely related data only in real time.

In order to improve the spatial resolution and spectral resolution of images, mainstream high-resolution optical satellites use different sensors to acquire high spatial resolution panchromatic images and low spatial resolution spectral images, respectively. The two focus on obtaining finer spatial textures and obtaining spectral features of ground objects, respectively. Compared with the direct acquisition of high-resolution multispectral images, this strategy can significantly reduce the cost of satellite development and data transmission pressure, so it is widely used; but in the later stage, it is necessary to fuse panchromatic images with multispectral images. It brings higher requirements for data processing and application. In the field of onboard real-time processing, research institutions around the world have carried out a series of explorations and achieved some achievements [24], [25], [26], [27], [28], [29], [30], [31], [32], [33], [34], [35], but, in general, they are still limited to processing with a small amount of data or general processing such as data compression and radiance processing that are easy to implement by hardware. In contrast, in panchromatic multispectral fusion processing, a series of processes such as data analysis, radiance correction, distortion correction, stitching, and overlapping region correction need to be performed on the separated panchromatic and multispectral images acquired by the camera first, to eliminate the radiation inconsistency and geometric distortion problems generated during satellite imaging. Then, pixel-level geometric registration of panchromatic multispectral images is required, and finally, pixel-level fusion can be performed. This processing has too many steps, has a large amount of input and output and temporary data, is difficult to implement by hardware. These characteristics determine that it is very difficult to realize real-time fusion processing based on the limited onboard conditions.

Under the current technical conditions, the hardware that can provide computing capabilities under limited onboard conditions is mainly based on embedded devices, such as DSP and FPGA [36], [37], [38]. This type of hardware realizes embedded processing coupled with software and hardware, but its versatility is poor, development is difficult, and it is difficult to efficiently and intelligently meet the current and future needs of various onboard time-sensitive remote sensing applications. In recent years, the emergence of general-purpose computing architectures based on embedded GPUs or AI chips has provided a feasible solution to the above problems. The computing performance of embedded GPUs or AI chips is significantly higher than that of DSP, and the power consumption is low, the hardware architecture is common and has the ability to provide high-performance computing capabilities in limited onboard conditions.

In view of the above problems, this article adopts embedded GPU as a low-power (TDP:15 W) onboard processing simulation system and, based on this system, studies the implementation approach of onboard real-time fusion processing. The main innovations include the following.

1) Propose a task-driven onboard fusion processing strategy to significantly reduce the amount of data and computing, and avoid I/O operations between internal and external memory.
2) Propose an object-space consistency-based registration method to avoid the computationally intensive and time-consuming pixel-level registration processing.
3) Formulate an efficient algorithm mapping strategy and deeply optimize the time-consuming algorithms to significantly shorten the processing time. On this basis, build a data extraction-processing-output parallel pipeline to achieve real-time stream computing.

II. Related Work

Since the input data of onboard fusion processing is the raw image separated by band and sensor, it is different from general fusion processing. In order to realize task-driven onboard fusion processing, it is necessary to accurately extract ROI data by real-time positioning the first time and perform radiance correction, sensor correction, band registration, stitching, and overlapping area correction on it, and on this basis, fusion processing can be performed. Among them, the real-time positioning of the ROI during the imaging process is a key step in onboard processing [39]. This section introduces the important algorithms involved in the above process.

A. Instant Geographic Locating

1) Strict Geometric Model: For optical linear array images, each line of the image is an independent central projection imaging result. By interpolating the information provided by the auxiliary data, the precise exterior orientation elements corresponding to each line, including position and attitude information, can be obtained.

The coordinates (X_s, Y_s, Z_s) [40] of the projection center of each image line at time t are interpolated from the GPS or BD2 data by a cubic polynomial model

$$\begin{cases} X_s\ (\bar{t}) = \ k_0 \ + k_1\bar{t} + \ k_2\bar{t}^2 + \ k_3\bar{t}^3 \\ Y_s\ (\bar{t}) \ = m_0 \ + m_1\bar{t} + m_2\bar{t}^2 + m_3\bar{t}^3 \\ Z_s\ (\bar{t}) = \ n_0 \ + n_1\bar{t} + \ n_2\bar{t}^2 + \ n_3\bar{t}^3 \end{cases} \tag{1}$$

where k_0, k_1, k_2, k_3, m_0, m_1, m_2, m_3 n_0, n_1, n_2, and n_3 are the coefficients of the cubic polynomial models that are fit by the observed data; $\bar{t}$ is the normalized imaging time, which is calculated as follows:

$$\bar{t} = \frac{t - t_{\text{start}}}{t_{\text{end}} - t_{\text{start}}} \tag{2}$$

where t_{start} and t_{end} are the start and end times of the satellite position data, respectively.

Similarly, the attitude information $(\varphi, \omega, \kappa)$ could be interpolated by a cubic polynomial model

$$\begin{cases} \varphi\ (\bar{t}) = \ k_0 \ + k_1\bar{t} + \ k_2\bar{t}^2 + \ k_3\bar{t}^3 \\ \omega\ (\bar{t}) \ = m_0 \ + m_1\bar{t} + m_2\bar{t}^2 + m_3\bar{t}^3 \\ \kappa\ (\bar{t}) = \ n_0 \ + n_1\bar{t} + \ n_2\bar{t}^2 + \ n_3\bar{t}^3 \end{cases} \tag{3}$$

where k_0, k_1, k_2, k_3, m_0, m_1, m_2, m_3 n_0, n_1, n_2, and n_3 are the coefficients of the cubic polynomial models that are fit by the observed data.

On this basis, the strict geometric model can be established as follows:

$$\begin{bmatrix} X - X_s\,(t) \\ Y - Y_s\,(t) \\ Z - Z_s\,(t) \end{bmatrix} = \lambda \cdot R_{\text{ECI}}^{\text{ECF}}\,(t) \cdot R_{\text{body}}^{\text{ECI}}\,(\varphi\,(t)\,, \omega\,(t)\,, \kappa\,(t))$$

$$\cdot\, R_{\text{sensor}}^{\text{body}} \begin{bmatrix} \tan\psi_x \\ \tan\psi_y \\ 1 \end{bmatrix} \tag{4}$$

where t is the imaging time of the scan line recorded by the camera; λ is a scale factor; ψ_x and ψ_y are the look angles; $R_{\text{body}}^{\text{sensor}}$ is the camera install matrix calculated by inflight calibration, which can be treated as a fixed value over a long period of time; $R_{\text{ECI}}^{\text{body}}(\varphi(t), \omega(t), \kappa(t))$ is the rotation matrix from the Earth-centered inertial (ECI) coordinate system to the satellite body coordinate system converted by rotation angles $\varphi(t), \omega(t)$, and $\kappa(t)$, which are the pitch, roll, and yaw angles, respectively, and interpolated from the attitude observation under the ECI coordinate system by time; $R_{\text{ECF}}^{\text{ECI}}(t)$ is the transformation matrix from the Earth-centered fixed (ECF) coordinate system to the ECI coordinate system; and $[X_s(t) \quad Y_s(t) \quad Z_s(t)]^T$ is the position vector of the projection center in the ECF coordinate system, interpolated from the position observation by time. Among them, the interior orientation elements ψ_x and ψ_y are described by the look angle of the camera coordinates; they are determined by laboratory calibration before launching and updated by inflight calibration during a certain period after launching [41], [42].

2) ROI Location: Based on the above strict geometric modeling method, the geographic coordinates corresponding to each pixel can be calculated. However, in the process of satellite imaging, it is impossible to complete the calculation of the positions of all image pixels and compare them with the position of the ROI because of the limitation of the computing resources on board. Therefore, the efficient ROI location algorithm is described as follows.

1) Establishing the strict geometric model of the current imaging line.
2) Calculating the geographic position of the first and last pixels of the current line at T_0, and obtaining the points (p_0, q_0).
3) Repeating steps 1 and 2 to obtain the next points (p_1, q_1) when $T_1 = T_0 \ + \Delta t$.
4) Determining whether the ROI center is located in the rectangle (p_0, q_0, p_1, q_1); if not, continue to repeat the above calculation steps after Δt time.
5) If the center of the ROI is located in the rectangle $(p_i, q_i, p_{i+1}, q_{i+1})$, calculate the exact image coordinates of the range of the ROI area.

Since only two points need to be calculated in a Δt timespan, the calculation can be completed in several milliseconds. The flowchart of this algorithm is shown in Fig. 1. After the locating is successful, an affine transformation model can be established according to the two corner points obtained in this calculation and the two corner points calculated last time

$$\begin{cases} s = a_0 \ + a_1 B + a_2 L \\ l = b_0 \ + b_1 B + b_2 L \end{cases} \tag{5}$$

where $a_0, a_1, a_2, b_0, b_1,$ and b_2 are the coefficients of the affine transformation model. They can be determined by the least-square method, and then, the pixel coordinates of the ROI center can be calculated, and the ROI data can be clipped for subsequent processing. ROI real-time positioning plays a very important role in reducing the amount of data and calculation and improving the real-time performance of onboard processing.

3) Rational Function Model: The strict geometric model establishes the mathematical relationship between the original image pixel coordinates and the geographic coordinates. However, there are two problems: 1) Each line of the image has independent imaging conditions, and it is necessary to establish similar but different strict models line by line; 2) the inverse calculation of the strict model from geographic coordinates to pixel coordinates requires iteration, which is computationally intensive and unstable. Therefore, after extracting the ROI region, this article uses a more general and efficient rational function model (RFM) to replace the strict model in the subsequent processing.

The RFM is a rational polynomial model that directly establishes the relationship between the geographic coordinates and the pixel coordinates [43], [44], [45]. The terrain-independent method can be used to convert the strict model to RFM with high accuracy without relying on any ground control points [46]: A sufficient number of virtual control points are generated using strict models first, and then, the rational polynomial coefficients (RPCs) of the high-precision RFM can be obtained by the least-square method.

The RFM performs mutual conversion calculations among pixel coordinates (l, s), geographic coordinates (B, L), and ellipsoid height H. (l_n, s_n) and (U, V, W) are the normalized

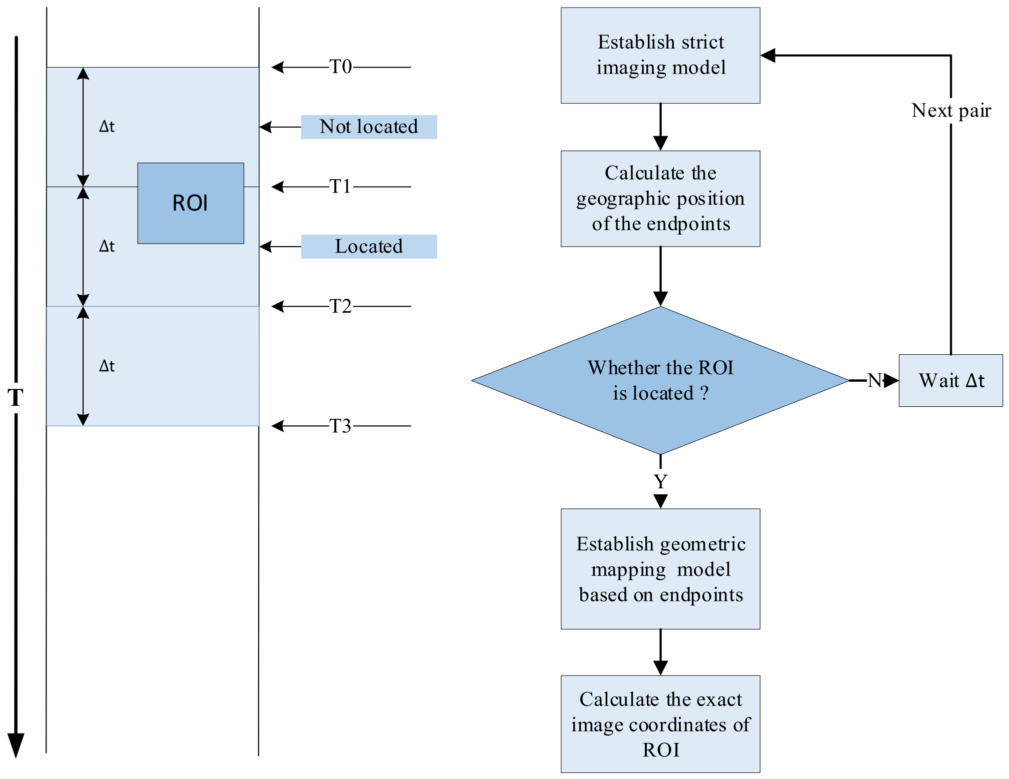

T0
Not located
T1
ROI
Located
T2
T3
Δt
Δt
Δt
T
Establish strict imaging model
Calculate the geographic position of the endpoints
Whether the ROI is located ?
N
Wait Δt
Next pair
Y
Establish geometric mapping model based on endpoints
Calculate the exact image coordinates of ROI

Fig. 1. Flowchart of ROI location [39].

coordinates of pixel coordinates (l, s) and geographic coordinates (B, L, H), respectively, and their relationship is

$$\begin{cases} l_n = \frac{\mathrm{Num}_L(U,V,W)}{\mathrm{Den}_L(U,V,W)} \\ s_n = \frac{\mathrm{Num}_S(U,V,W)}{\mathrm{Den}_S(U,V,W)} \end{cases} . \tag{6}$$

The polynomial numerator and denominator are, respectively, expressed as follows. Among them, a_i, b_i, c_i, d_i $(i = 1, 2, \ldots, 20)$ are coefficients of RFM, which have been solved when the model is generated, Equation (7) shown at the bottom of this page

B. Panchromatic Multispectral Fusion

After extracting, the original panchromatic and multispectral data covering the ROI area, sensor correction, fusion, and geometric correction are required for both of them to obtain the high-resolution multispectral fusion result with geographic information.

1) Sensor Correction: Sensor correction is currently the prerequisite step for photogrammetric processing and application of high-resolution optical satellite data that mainly solves the problem caused by camera design and imaging characters, such as image distortion caused by lens distortion and an unstable platform, sensor image stitching, and multispectral image band-to-band registration. The standard image product with an RFM can be obtained after sensor correction.

Generally, the current sensor correction method can be categorized into two classes: the image-space-oriented (ISO) method, and the object-space-oriented (OSO) method [47]. Unlike the ISO method that mainly solves the problem of sensor image stitching and band-to-band registration based on image matching

$$\begin{cases} \mathrm{Num}_L\ (U,V,W) = \begin{array}{c} a_1 + a_2V + a_3U + a_4W + a_5VU + a_6VW + a_7UW + a_8V^2 \\ +a_9U^2 + a_{10}W^2 + a_{11}VUW + a_{12}V^3 + a_{13}VU^2 + a_{14}VW^2 \\ +a_{15}V^2U + a_{16}U^3 + a_{17}UW^2 + a_{18}V^2W + a_{19}U^2W + a_{20}W^3 \end{array} \\ \mathrm{Den}_L\ (U,V,W) = \begin{array}{c} b_1 + b_2V + b_3U + b_4W + b_5VU + b_6VW + b_7UW + b_8V^2 \\ +b_9U^2 + b_{10}W^2 + b_{11}VUW + b_{12}V^3 + b_{13}VU^2 + b_{14}VW^2 \\ +b_{15}V^2U + b_{16}U^3 + b_{17}UW^2 + b_{18}V^2W + b_{19}U^2W + b_{20}W^3 \end{array} \\ \mathrm{Num}_S\ (U,V,W) = \begin{array}{c} c_1 + c_2V + c_3U + c_4W + c_5VU + c_6VW + c_7UW + c_8V^2 \\ +c_9U^2 + c_{10}W^2 + c_{11}VUW + c_{12}V^3 + c_{13}VU^2 + c_{14}VW^2 \\ +c_{15}V^2U + c_{16}U^3 + c_{17}UW^2 + c_{18}V^2W + c_{19}U^2W + c_{20}W^3 \end{array} \\ \mathrm{Den}_S\ (U,V,W) = \begin{array}{c} d_1 + d_2V + d_3U + d_4W + d_5VU + d_6VW + d_7UW + d_8V^2 \\ +d_9U^2 + d_{10}W^2 + d_{11}VUW + d_{12}V^3 + d_{13}VU^2 + d_{14}VW^2 \\ +d_{15}V^2U + d_{16}U^3 + d_{17}UW^2 + d_{18}V^2W + d_{19}U^2W + d_{20}W^3. \end{array} \end{cases} \tag{7}$$

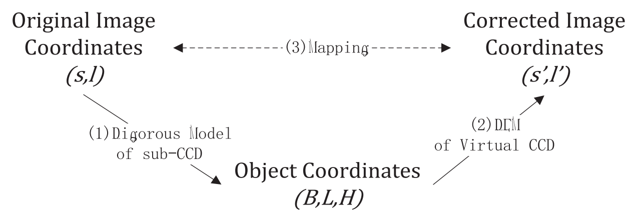

Original Image Coordinates
(s,l)
(3)Mapping
Corrected Image Coordinates
(s',l')
(1)Digorous Model of sub-CCD
(2)DCM of Virtual CCD
Object Coordinates
(B,L,H)

Fig. 2. OSO sensor correction method.

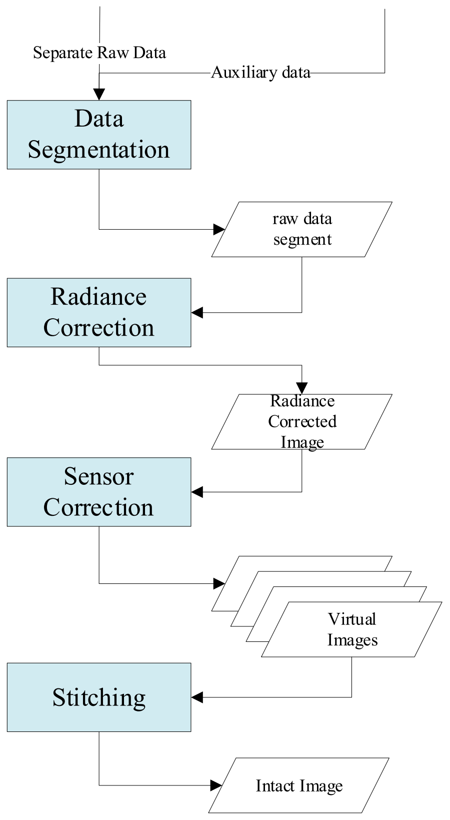


Fig. 3. Flowchart of sensor correction.

[48], the OSO method establishes the corresponding relationship between the original image and the corrected image based on the image geometric model with the object space as an intermediary, and it is theoretically stricter and more desirable for high-precision image processing and application (as shown in Fig. 2). In the OSO method, a virtual sensor is used to replace the original physical sensor with multiple bands [49], and furthermore, a virtual steady reimaging model based on the virtual sensor is used to correct the distortion caused by both the camera and satellite [50], [51]. Therefore, this article adopts the OSO method to solve the geometric problem of onboard fusion processing. The procedure of sensor correction is shown in Fig. 3.

2) Smoothing Filter-Based Intensity Modulation: Mainstream optical satellites usually provide panchromatic images with high spatial resolution and multispectral images with small data volumes. After fusion processing, the two can obtain high-resolution multispectral images and enhance the use value of data products. Affected by the limitations of onboard processing and real-time requirements, the algorithm suitable for onboard processing should have the characteristics of a small amount of calculation, no dependence on external memory, and suitable for block-independent processing. Among many fusion algorithms, smoothing filter-based intensity modulation (SFIM) is a high-fidelity image fusion algorithm that can well preserve spectral features [52], [53]. It can divide the image to be processed into multiple blocks for parallel computing, which is suitable for onboard processing. The principle is as follows.

Suppose the imaging band of the remote sensing image is λ, and its pixel value is determined by the irradiance $E(\lambda)$ and the surface reflectance $\rho(\lambda)$ [54]

$$DN\ (\lambda) = \ \rho\,(\lambda)\,E\,(\lambda)\,. \tag{8}$$

The SFIM is defined as

$$\begin{aligned} DN(\lambda)_{\text{sim}} &= \frac{DN(\lambda)_{\text{low}} DN(\gamma)_{\text{high}}}{DN(\gamma)_{\text{low}}} \\ &= \frac{\rho(\lambda)_{\text{low}} E(\lambda)_{\text{low}} \rho(\gamma)_{\text{high}} E(\gamma)_{\text{high}}}{\rho(\gamma)_{\text{low}} E(\gamma)_{\text{low}}}\,. \end{aligned} \tag{9}$$

For simultaneous imaging of panchromatic and multispectral data, it can be considered that the solar radiation at the imaging moment is the same, and for images with the same resolution, it can be assumed that $E(\lambda) \approx E(\gamma)$ [55]; the surface reflectance of the same object is also the same, it can be assumed that $\rho_{\text{low}} \approx \rho_{\text{high}}$. $E(\lambda)_{\text{low}}$ and $E(\gamma)_{\text{low}}$, $\rho(\gamma)_{\text{low}}$, and $\rho(\gamma)_{\text{high}}$ in (9) can cancel each other out, then

$$\begin{aligned} DN(\lambda)_{\text{sim}} &\approx \rho(\lambda)_{\text{low}} E(\lambda)_{\text{high}} \approx \rho(\lambda)_{\text{high}}\ E(\lambda)_{\text{high}} \\ &= DN(\lambda)_{\text{high}}. \end{aligned} \tag{10}$$

It shows that the value obtained by the fusion model defined by (9) is approximately equal to the real value.

Therefore, for the panchromatic multispectral fusion, we can get

$$\text{Fusion} \ = \frac{\text{MS} \times \text{PAN}}{\text{PAN}'} \tag{11}$$

where MS represents a multispectral image, PAN represents a panchromatic image, and $\text{PAN}^{'}$ represents a low-resolution panchromatic image, Fusion represents a fusion result. $\frac{\text{PAN}}{\text{PAN}'}$ is considered to preserve the edge detail information of high-resolution images while basically eliminating spectral and contrast information.

The SFIM model can be regarded as adding high-resolution details to low-resolution images with good color retention. The key to its fusion effect is the generation of degraded panchromatic images. Therefore, in this article, the Gaussian filter is used instead of the neighborhood mean filter in the original SFIM to degrade the panchromatic image, in order to obtain a degraded panchromatic image closer to the multispectral image through parameter adjustment, and then, obtain a more accurate $\frac{\text{PAN}}{\text{PAN}'}$.

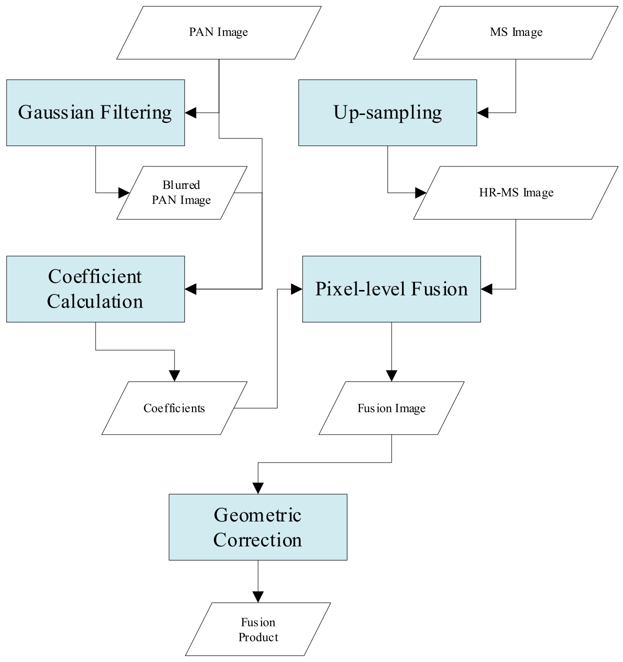


Fig. 4. Flowchart of SFIM fusion.

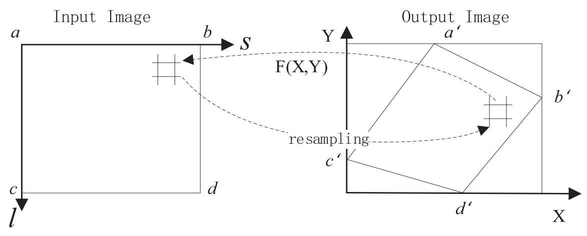


Fig. 5. Indirect correction method.

The Gaussian filter can be described as

$$G\ (x, y) = \frac{1}{2\pi\sigma^2}\, e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{12}$$

where (x, y) is the position in the convolution template, $G(x, y)$ is the value of the position. The filter effect can be adjusted by changing the value of σ. The flowchart is shown in Fig. 4.

3) Geometric Correction: Geometric correction is the process of geometrically transforming sensor-corrected images according to an appropriate model and geocoding the image [56]. The commonly used geometric correction methods include the direct correction method and the indirect correction method, as shown in Fig. 5.

The indirect correction method establishes the relationship between the output image and the input image, as shown in (13). For each pixel (X, Y) on the output image, its coordinates (s, l) on the input image are calculated according to F_s and F_l, then resampling is performed according to the value of surrounding pixels. Usually, the nearest neighbor method, bilinear interpolation method, and bicubic interpolation method can be used

$$\begin{cases} s\ = F_s\ (X, Y) \\ l\ = F_l\ (X, Y) \end{cases}. \tag{13}$$

The coordinate transformation model F_s and F_l could be a polynomial transformation model, strict geometric model, or RPC model.

III. Proposed Approach

In the traditional data-centric processing method, the processing steps are clearly divided. Each step of the algorithm only needs to focus on the processing of independent operations, focus on the input data, and make the output data the best possible effect. Its disadvantage is that the algorithm of each step is relatively fragmented, and the overall efficiency is low. In order to realize real-time processing under the condition of limited resources, it is necessary to consider and optimize the algorithm processing flow in order to improve the algorithm processing efficiency as much as possible.

A. Task-Driven Onboard Fusion Processing

1) Strategy of Task-Driven Processing: Task-driven onboard processing performs instant geographic locating, filtering, and processing of raw data according to the task instructions uploaded by ground users. It needs to be completed in real-time under limited onboard conditions, which is more complex and difficult than traditional processing.

As shown in Fig. 6, the preset or annotation parameters that need to be used in the task-driven processing include camera geometry calibration parameters, camera radiance calibration parameters, Earth rotation parameters (precession, nutation, pole shift), algorithm configuration parameters, and global digital elevation model (DEM). The flow-in raw data come from the satellite imaging system, including raw images and imaging auxiliary data; the attitude and position data are broadcasted through the satellite platform bus. The processing steps include data analysis, instant geometric positioning, ROI extraction, relative radiance correction, sensor correction, fusion, and geometric correction. The final output is the geometrically corrected fused image of the ROI specified by the task instruction.

According to the ROI position specified in the task instruction, combined with instant geometric positioning calculations, the important data can be accurately positioned as quickly as possible when the onboard processing platform is unable to continuously cache the whole flow-in data. And only the data in this area are processed, thereby significantly reducing the overall calculation amount and improving the processing timeliness. The key lies in the high-precision geometric positioning completed in real time during the data flow-in process.

2) Parallel Progressive Fusion Algorithm: In order to make full use of the embedded GPU for parallel processing, the fusion algorithm described is been adjusted: First, calculate the overlapping range of sensor-corrected panchromatic and multispectral image and divide it into small blocks; then, perform fusion processing of each block in memory, including Gaussian filtering of panchromatic image, fusion coefficient calculation, upsampling of multispectral image, and pixel-level fusion; merge the fusion results of each block, and perform the geometric correction to get the final product. On the one hand, the improved algorithm makes better use of the parallelizability of the SFIM method, and

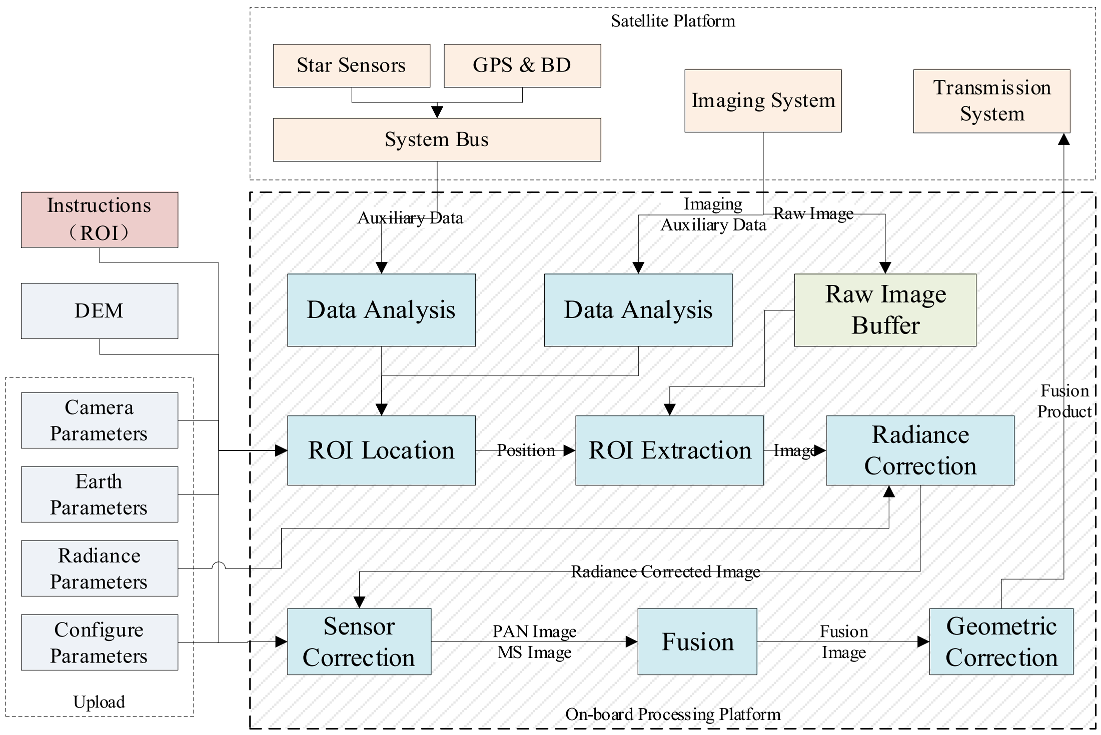


Fig. 6. Strategy of task-driven processing.

TABLE I
ANALYSIS OF OCCUPIED MEMORY SPACE

	Input Data	Filtered Data	Fusion Coefficients	Upsampling	Fusion Result	Sum
Original method	$1.25C$	$1C$	$1C$	$4C$	$4C$	$11.25C$
Parallel method		whk	whk	$4whk$	$4whk$	$1.25C+10whk$

can adapt and adjust the algorithm by flexibly setting the grid size and the number of parallel blocks, while avoiding memory overflow caused by too large temporary data; on the other hand, the I/O operations between the steps of the original algorithm are omitted, which can significantly speed up the overall processing time. The adjusted algorithm is shown in Fig. 7.

Let W_{pan}, H_{pan}, W_{ms}, and H_{ms} be the width and height of the panchromatic and multispectral image respectively, and there is a 4:1 relationship between them. w, h are the width and height of a single block, respectively, and k is the number of simultaneous parallel blocks. For the convenience of discussion, let $C = W_{\mathrm{pan}} \times H_{\mathrm{pan}}$, then the occupied memory space analysis is as given in Table I.

From the above analysis, it can be seen that if it is divided into 20×20 grids and the parallelism is set to 10, the memory capacity occupied by the method in this article is $1.5C$, which is much smaller than the $11.25C$ of the original method, which can better adapt to the onboard processing condition.

B. Object-Space-Consistency-Based Registration

Traditional fusion processing requires high-precision registration of the input panchromatic and multispectral images to keep them geometrically consistent. However, the registration process is computationally intensive and time-consuming, and onboard applications should be avoided as much as possible. Considering that the panchromatic and multispectral images are almost simultaneously imaged, there are relatively consistent imaging conditions, and theoretically, the registration of the two can be completed through sensor correction.

However, in the OSO sensor correction method, in order to ensure reimaging accuracy, the ideal virtual sensor needs to be set as close as possible to multiple physical sensors. According to this principle, the virtual sensor is usually set in the middle position of physical sensors. Therefore, the difference in look angle between virtual sensor and physical sensors could be as small as possible, so as to keep the geometric positioning accuracy of the virtual sensor and multiple physical sensors as consistent as possible. Therefore, the set panchromatic virtual sensor and the multispectral virtual sensor cannot overlap on the camera focal plane, resulting in a certain deviation in the geometric position of the same ground object obtained after the two perform sensor correction respectively, so it cannot be fusion directly. As shown in Fig. 8, green represents the panchromatic physical sensors (solid line) and the virtual sensor (dotted line), and blue represents the multispectral physical sensors (solid

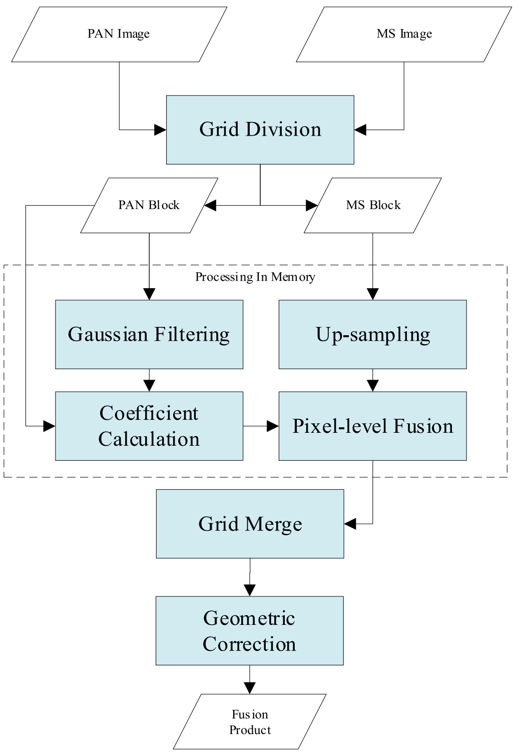


Fig. 7. Parallel progressive fusion algorithm.

line) and the virtual sensor (dotted line). It can be seen that the virtual panchromatic sensor does not overlap with the virtual multispectral sensor in the camera focal plane.

Therefore, in order to fuse directly from the panchromatic and multispectral sensor correction results, it is necessary to fine-tune the settings of virtual sensors, to make the two overlaps in the camera focal plane. The look angle of every sensor pixel can be described as [46]

$$\begin{cases}\Psi_x\ (s) = a_0\ + a_1 s + a_2 s^2 + a_3 s^3 \\ \Psi_y\ (s) = b_0\ + b_1 s + b_2 s^2 + b_3 s^3\end{cases} \tag{14}$$

where a_0, a_1, a_2, a_3, b_0, b_1, b_2, and b_3 are polynomial coefficients, s represents the pixel index number of the sensor. A complete set of parameters represents a piece of physical or virtual sensor on the camera focal plane. According to the parameters, the look angle of each pixel on the sensor can be obtained. For an ideal virtual sensor without distortion, the other parameters except a_0, b_0, and b_1 can be set to 0, where a_0 and b_0 represent the position of the first pixel from the left of the sensor on the camera focal plane, and b_1 indicates the size of the pixel.

For the convenience of discussion, we introduce the superscript mark on the coefficient to identify the sensor and take a panchromatic virtual sensor as an example. As long as a_0^{PAN}, b_0^{PAN}, and b_1^{PAN} are set correctly, the ideal panchromatic virtual sensor is set correctly. It can be set as follows through the parameters of the physical sensors

$$\begin{cases} a_0^{\text{PAN}} = a_0^{\text{PAN1}} \\ b_0^{\text{PAN}} = \frac{\sum_{i=1}^{4} b_0^{\text{PAN}i}}{4} \\ b_1^{\text{PAN}} = \frac{\sum_{i=1}^{4} b_1^{\text{PAN}i}}{4}\end{cases}. \tag{15}$$

Similarly, the parameters of multispectral virtual sensor can be obtained (j identifies the band index number)

$$\begin{cases} a_0^{\text{MS}} = a_0^{\text{MS1}} \\ b_0^{\text{MS}} = \frac{\sum_{i=1,j=1}^{4,4} b_0^{\text{MS}ij}}{16} \\ b_1^{\text{MS}} = \frac{\sum_{i=1,j=1}^{4,4} b_1^{\text{MS}ij}}{16}\end{cases}. \tag{16}$$

On this basis, in order to make the panchromatic virtual sensor and the multispectral virtual sensor overlap on the camera focal plane and to consider the impact of the new virtual sensor settings on panchromatic and multispectral physical sensors, a_0 and b_0 of virtual sensors can be adjusted according to (15) and (16)

$$\begin{cases} a_0^{\text{Final}} = \left(a_0^{\text{PAN}} + a_0^{\text{MS}}\right)/2 \\ b_0^{\text{Final}} = \left(b_0^{\text{PAN}} + b_0^{\text{MS}}\right)/2\end{cases}. \tag{17}$$

Then, set a_0 and b_0 of the panchromatic virtual sensor and the multispectral virtual sensor to the values obtained by (17), and set b_1 of the multispectral virtual sensor to four times b_1 of the panchromatic virtual sensor, namely: $b_1^{\text{MS}} = b_1^{\text{PAN}}\ \times 4$.

Furthermore, due to differences in panchromatic multispectral pixel size and integration time, the positional relationship between the actual panchromatic and multispectral virtual sensors is shown in Fig. 9. Green represents panchromatic virtual pixels and blue represents multispectral virtual pixels. It is necessary to make fine-tuning to the multispectral virtual sensor, that is, to move 1.5 panchromatic pixels to the right and down to move it to the red position

$$\begin{cases} a_0^{\text{MS}} = a_0^{\text{Final}}\ - 1.5 \times b_1^{\text{PAN}} \\ b_0^{\text{MS}} = b_0^{\text{Final}}\ + 1.5 \times b_1^{\text{PAN}}\end{cases}. \tag{18}$$

It should be noted that when fine-tuning, it is necessary to pay attention to the definition of the direction of the x and y axes of the camera focal plane, and the sign of b_1^{PAN}, which is negative here.

The adjusted panchromatic and multispectral virtual sensors overlap at the camera focal plane, making it possible to directly fuse the images after sensor correction.

C. *Algorithm Mapping Strategy*

In order to fully adapt to the characteristics of the onboard embedded GPU architecture and improve the use efficiency, it is first necessary to analyze the characteristics of the algorithm. Then, specify the overall algorithm mapping strategy and optimize the time-consuming algorithm as much as possible to pursue real-time performance.

1) Algorithm Analysis and Overall Mapping Strategy: The onboard fusion processing involved in this article mainly includes four steps: 1) data analysis, 2) ROI extraction, 3) sensor correction, and 4) fusion and geometric correction. Among them, the main operation of data analysis is receiving, and the calculation amount can be ignored, so it will not be discussed later.

Table II analyzes the 4 steps and 17 types of algorithms involved in this article from two aspects: 1) computing load

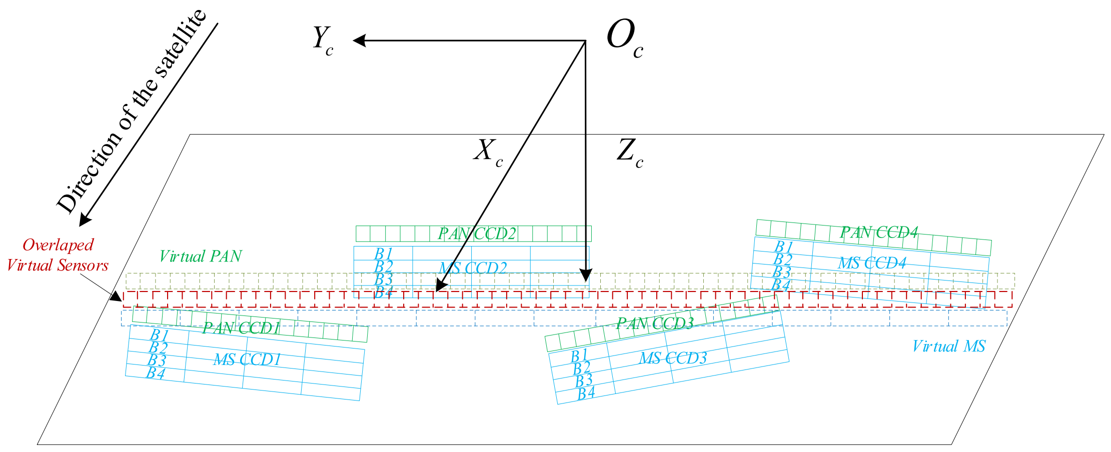


Fig. 8. Sensors on camera focal plane.

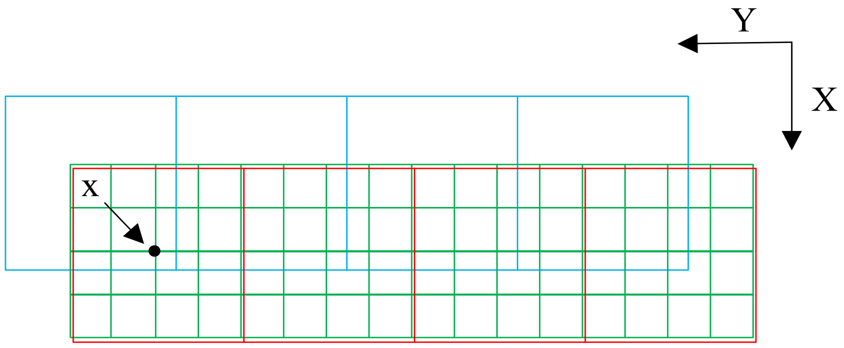


Fig. 9. Fine-tuning of multispectral virtual sensor.

TABLE II
CHARACTERISTICS OF ALGORITHM

Step	Algorithm	Computing Load	Parallelizability
Data Analysis	Data analysis	Light	Low
ROI Extraction	Attitude Interpolation	Light	Low
	Orbit Interpolation	Light	Low
	Strict Modeling	Light	Low
	ROI Location	Light	Low
	Radiance Correction	Light	High
Sensor Correction	Attitude Interpolation	Light	Low
	Orbit Interpolation	Light	Low
	Strict Modeling	Light	Low
	RPCs Calculation	Light	Middle
	Resampling	Heavy	High
	Stitching	Light	Low
Fusion and Geometric Correction	Gaussian Filtering	Heavy	High
	Coefficient Calculation	Medium	High
	Multispectral Upsampling	Heavy	High
	Pixel-level Fusion	Medium	High
	Geometric Correction	Heavy	High

and 2) parallelizability. In general, algorithms with light computing load or low parallelizability are mapped to the CPU cores for processing, including attitude interpolation, orbit interpolation, strict modeling, ROI location, radiance correction; algorithms with a slightly higher computing load, including RPCs calculation and image stitching, only need to be run once, and mapping them to the CPU for multicore parallel processing can meet the requirements well; algorithms that require pixel-by-pixel processing, with heavy computing load and high parallelizability are compatible with the large-scale parallel

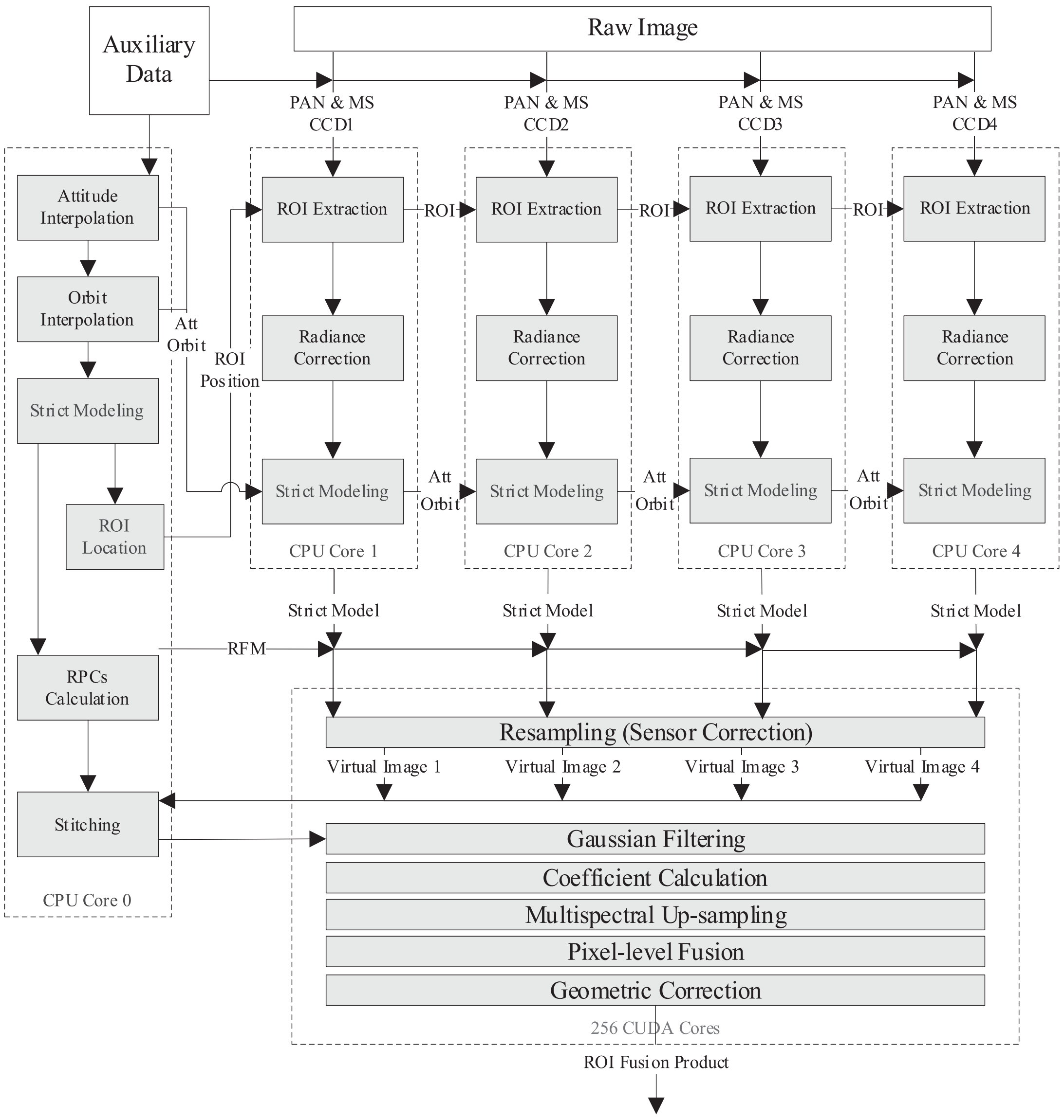


Fig. 10. Overall mapping strategy.

architecture of the GPU, and are mapped to the GPU for parallel processing, including resampling, Gaussian filtering, coefficient calculation, multispectral upsampling, pixel-level fusion, and geometric correction. The overall mapping strategy is shown in Fig. 10.

The embedded GPU module Terga X2 used in this article contains six CPU cores. For the satellite data including four linear array sensors in this article, a parallel processing strategy of four physical sensors and one virtual sensor can be used. The processing tasks related to each sensor are handled by 1 CPU core. The remaining 1 CPU core is responsible for processing other auxiliary computing tasks, such as data exchange, thread scheduling, system services, etc. When the system is running, each sensor starts an independent thread for processing, and the operating system of Terga X2 will align and schedule to ensure that they can be allocated to their respective CPU cores. For algorithms such as resampling, Gaussian filtering, coefficient calculation, multispectral upsampling, pixel-level fusion, and geometric correction that involve a large number of pixel-by-pixel operations, 256 CUDA cores of the embedded GPU are used for parallel processing.

Considering the processing flow, assume that the first CPU core (recorded as CPU core 0 in Fig. 10) starts to execute the main process: According to the continuously flow-in attitude data and orbit data, using the parameters of the virtual sensor, the first core performs attitude interpolation, orbit interpolation, strict modeling, and ROI location to determine whether the current push-broom imaging range covers the center point of the ROI. Once it is successfully covered, the accurate range of the ROI area is further calculated, and the rest of the CPU cores (CPU cores 1–4), respectively, perform cropping and radiance correction on the four physical sensor images according to the obtained range and, then, perform strict modeling. At the same time, CPU core 0 calculates the RPCs of the ROI area. Using the RPCs obtained by CPU core 0 and the strict model constructed by CPU cores 1–4, the pixel-by-pixel mapping

TABLE III
PSEUDOCODE OF SENSOR CORRECTION

```
Sensor Correction
Input: raw image and strict model of each sensor, RFM, DEM
Output: virtual image
1: Establish the corresponding relationship between raw image pixels and virtual image pixels
     $(s,l)_{ori} \leftrightarrow (s',l')_{vir}$
2: Pixel resampling (CUDA Parallelism)
  2.1 Divide the image into grids, each grid contains $(m \times n)$ pixels
  2.2 Specify CUDA thread parameters (closely related to performance) , including:
      1) The number of blocks in each CUDA grid: $(B_x, B_y)$;
      2) The number of threads in each CUDA block: $(T_x, T_y)$;
      3) The number of pixels processed in each CUDA thread: $(P_x, P_y)$.
      Make sure that $(B_x \times T_x \times P_x, B_y \times T_y \times P_y) \geq (m, n)$.
  2.3 $(B_x \times T_x, B_y \times T_y)$ threads in each CUDA grid are simultaneously submitted for parallel
       execution:
   For each image grid $(m,n)$, submit $(B_x \times T_x, B_y \times T_y)$ threads
        2.3.1 Each thread processes $(P_x, P_y)$ pixels
      Every Pixel
            2.3.1.1 Calculate the corresponding original coordinates $(s,l)_{ori}$
            2.3.1.2 Obtain adjacent pixels around the coordinate $(s,l)_{ori}$
            2.3.1.3 Calculate temporary variables for interpolation
            2.3.1.4 Calculate output pixel value
      End pixel loop
   End grid loop
```

relationship from the original physical image to the virtual image can be obtained. According to this mapping relationship, the resampling operation is performed in parallel on the 256 CUDA cores of the GPU to obtain virtual images. Then, the CPU core 0 stitches the virtual images to obtain the whole panchromatic and multispectral virtual image of the ROI area. Furthermore, algorithms such as Gaussian filtering, coefficient calculation, multispectral upsampling, pixel-level fusion, and geometric correction are performed on the 256 CUDA cores of the GPU to complete fusion processing and obtain the final high-resolution multispectral product of the ROI.

2) Time-Consuming Algorithm Mapping Strategy: Time-consuming algorithms that require pixel-by-pixel processing with heavy computing load and high parallelizability consume most processing time of the system and are the bottleneck of the system. Effectively improving their performance plays a decisive role in improving the overall performance of the onboard processing. Their mapping strategies are discussed as follows.

a) Sensor correction: The final step in the sensor correction process is pixel-by-pixel resampling. First, according to the physical sensor strict model and virtual sensor RPCs model obtained in the previous steps, the pixel-by-pixel correspondence between the original images and the virtual image in the ROI area can be obtained; on this basis, the corresponding position of any pixel of the virtual image on the original image can be obtained; then according to the original value of the adjacent points on the original image, the virtual image pixel can be obtained by interpolation. The pseudocode of the algorithm is shown in Table III, and the CPU/GPU collaboration process of the algorithm is shown in Fig. 11.

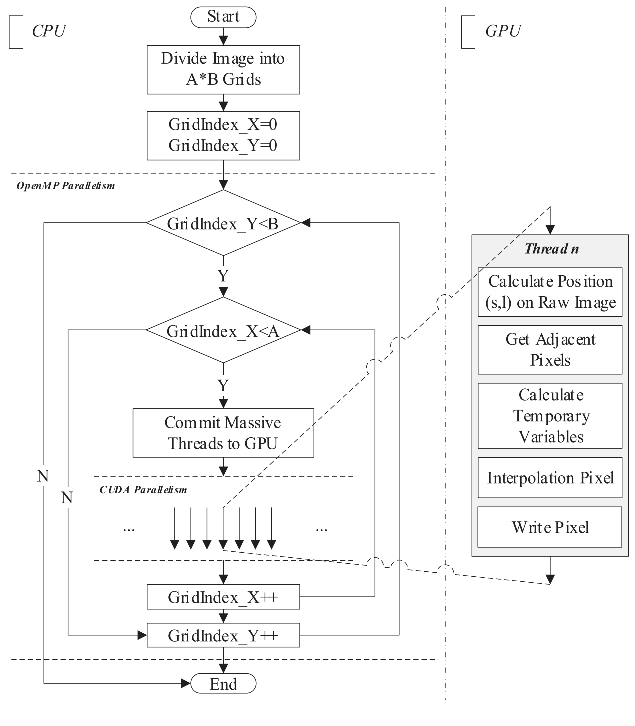


Fig. 11. CPU/GPU collaboration process of sensor correction.

b) Fusion and geometric correction: Fusion and geometric correction process the stitched panchromatic and

TABLE IV
PSEUDOCODE OF FUSION AND GEOMETRIC CORRECTION

```
Fusion and Geometric Correction
Input: panchromatic, multispectral virtual image, RFM, DEM
Output: ROI fusion product
1:  Fusion processing: (CUDA Parallelism)
  1.1 Divide the image into grids, each grid contains $(m \times n)$ pixels
  1.2 Specify CUDA thread parameters (closely related to performance), including:
        1) The number of blocks in each CUDA grid: $(B_x, B_y)$;
        2) The number of threads in each CUDA block: $(T_x, T_y)$;
        3) The number of pixels processed in each CUDA thread: $(P_x, P_y)$.
        Make sure that $(B_x \times T_x \times P_x, B_y \times T_y \times P_y) \geq (m, n)$.
  1.3 $(B_x \times T_x, B_y \times T_y)$ threads in each CUDA grid are simultaneously submitted for parallel
        execution:
    For each panchromatic image grid $(m, n)$, submit $(B_x \times T_x, B_y \times T_y)$ threads
    |     2.3.1 Each thread processes $(P_x, P_y)$ pixels
    | Every Pixel
    | |     2.3.1.1 Calculate the value of the blurred panchromatic pixel
    | |     2.3.1.2 Calculate edge information
    | |     2.3.1.3 Interpolation to obtain the upsampled multispectral pixel
    | |     2.3.1.4 Fusion of edge information and multispectral pixel to obtain fusion
    | | pixel
    | End pixel loop
    End grid loop
2:  Pixel resampling (refer to sensor correction)
```

multispectral virtual images of the ROI area. First, the blurred panchromatic image is obtained through Gaussian filtering; on this basis, the edge information of the panchromatic image is extracted through the ratio of the original panchromatic image and the blurred panchromatic image; then, according to the resolution of the original panchromatic image, the upsampling is performed on the multispectral image; then, fuse the extracted edge information with the upsampled multispectral image; finally, according to the corresponding relationship between the pixels of the fusion product and their geographical location, resampling is performed on a band-by-band and pixel-by-pixel to obtain the geometrically corrected fused image of ROI. The pseudocode of the algorithm is shown in Table IV, and the CPU/GPU collaboration process of the algorithm is shown in Fig. 12.

3) Pipeline Parallelism: The execution of any computer algorithm includes three major steps: 1) data input, 2) data processing, and 3) data output. For the onboard fusion processing, the data to be processed are remote sensing images, and the amount of data itself is large. Compared with the time-consuming data processing step, the data input and data output steps are also time-consuming and cannot be ignored. Fortunately, in a computer system, data input and output are executed by communication interfaces or external memory while processing is executed by the CPU cores and internal memory, which are different components. Therefore, considering that the Tegra X2 module is equipped with an operating system based on a complete Linux kernel, it has the ability to efficiently manage computing resources, storage resources, and communication interfaces. Using the multitasking capability of the Tegra X2 module, parallel pipelines can be built. The data input, data processing, and data output can be regarded as three parallel steps of the pipeline, thereby hiding time-consuming and improving overall efficiency. In fact, under appropriate scenarios and settings, this strategy can be applied not only within a single computing node but also in the collaboration of multiple computing nodes.

Fig. 13 shows the principle of building a parallel pipeline between multiple computing nodes and within a computing node: the data source sends data to each processing node continuously; and each processing node automatically segments the data, and internally parallelize the input, processing, and output steps; at the same time, the processing nodes are also parallelized through data segments. In order to compare the time-consuming difference before and after the parallel pipeline, let t_0, t_1, and t_2 be time consumption of input, processing, and output, and the number of data segments is n, the number of nodes is m, then, in general, the processing time t_s equals

$$t_s = (m-1) \times t_0 + \sum_{i=0}^{2} t_i \times \frac{n}{m} \tag{19}$$

in pipeline parallel mode, the processing time t_p equals

$$\begin{cases} t_p = (m-1) \times t_0 + \left(\sum_{\substack{i=0,\\ i \neq \max}}^{2} t_i + \frac{n}{m} t_{\max} \right) \\ t_{\max} = \max(t_0, t_1, t_2) \end{cases} . \tag{20}$$

Theoretically, if the data segments are divided finely enough, then there is $n \to \infty$. At this time, under the same input data and the same number of processing nodes, the parallel acceleration

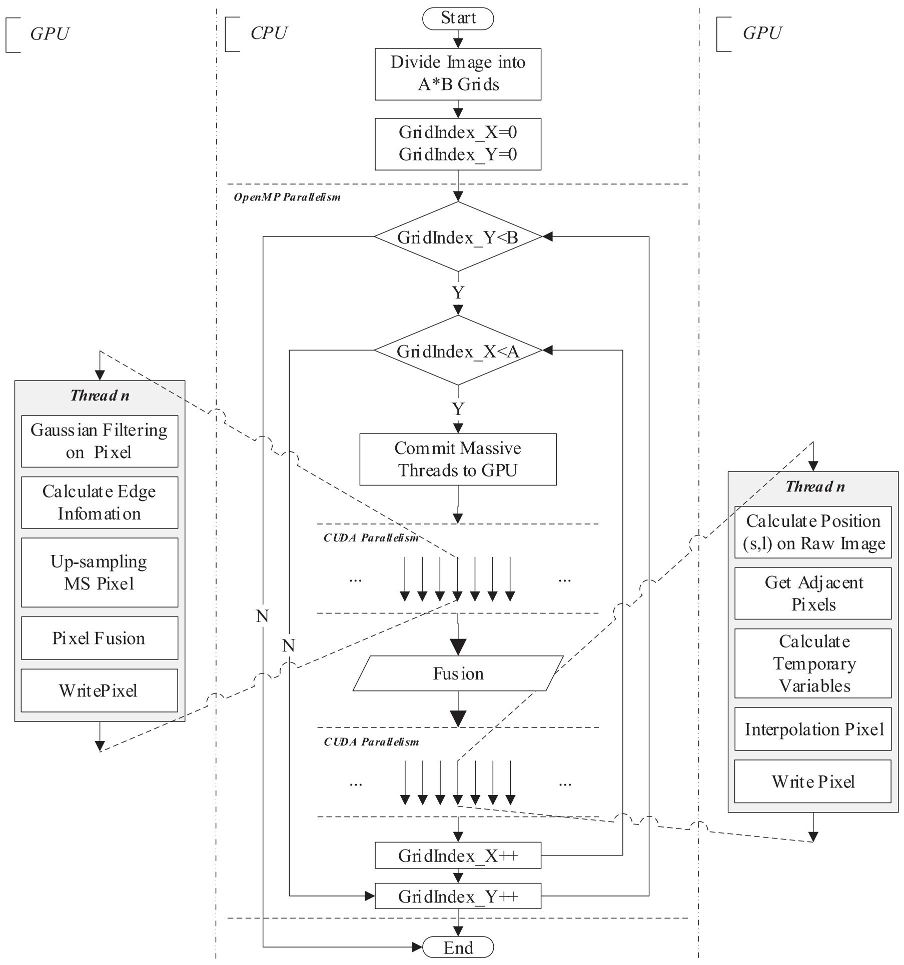


Fig. 12. CPU/GPU collaboration process of fusion and geometric correction.

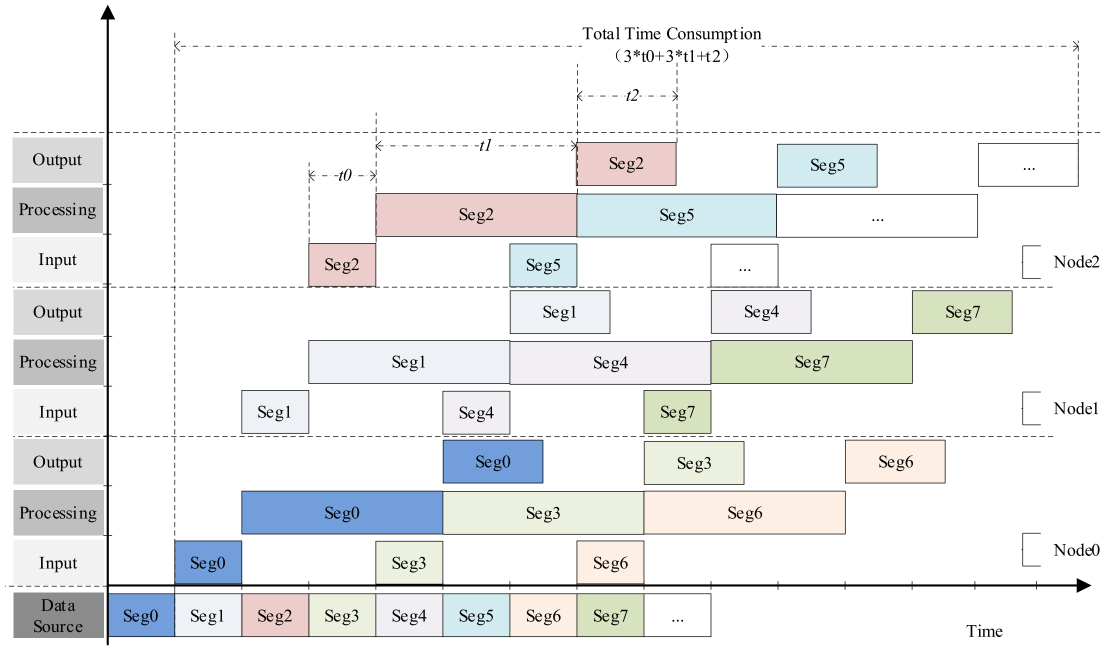


Fig. 13. Timeline of a parallel pipeline.

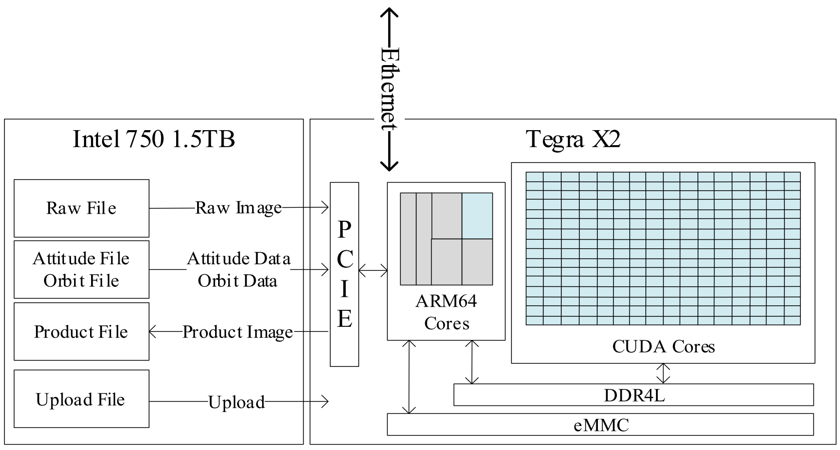


Fig. 14. Hardware architecture of simulation platform.

ratio of the pipeline is

$$\lim_{n\to\infty}\left(\frac{t_s}{t_p}\right) = 1 + \frac{\sum^{2}_{i\,=\,0,\; i\neq \max} t_i}{t_{\max}}. \quad (21)$$

Equation (21) shows the following.

1) Dividing finer data segments is more conducive to improving the performance of parallel pipelines.
2) Ideally, when $t_0 = t_1 = t_2$, the processing performance can be increased up to three times.
3) For the case of only one processing node in this article, the value of m is 1, which does not affect the above conclusions.

It should be noted that if t_1 or t_2 is significantly bigger than t_0, it means that not all input data can be processed in time. At this time, either adjust the algorithm to reduce t_1, t_2, or periodically discard part of the flow-in data to meet the stream processing constraints. Under ideal circumstances, the pipeline can significantly improve the overall efficiency of the system, but the conditions for maintaining its stability in actual scenarios are relatively harsh. In actual processing, affected by dynamic factors such as data characteristics, singular values, and thread scheduling delays, the processing time of similar tasks fluctuates, which may cause local delays and waits, and such delays will accumulate. In severe cases, the pipeline will be seriously delayed or even crushed. In order to alleviate this situation, usually enough buffer is set between parallel steps or enough performance margin is left when setting up, but the former will waste more system memory, and the latter will reduce the overall timeliness. Therefore, whether to adopt this strategy needs to be analyzed in detail according to the actual measurement situation.

IV. Experiments

A. Experiment Design

1) Simulation Onboard Platform: In this article, referring to the actual onboard embedded GPU processing hardware architecture, without affecting the performance of the core embedded GPU processing system, commercial hardware is used to build an alternative platform as shown in Fig. 14 to examine the performance of the proposed approach.

The experimental system uses the nVidia Tegra X2 (TDP:15 W) as the processing core, and the Intel 750 1.5 TB solid drive to replace the FPGA in the real onboard system, and replaces the interaction between the FPGA and the satellite system by reading and writing files, including platform system, imaging system, data transmission system, and uploading system. The choice of the hardware architecture of the experimental system is mainly based on the following considerations.

a) 4x PCIE 2.0 bandwidth can reach 32 Gb/s, which is the highest speed interface supported by Tegra X2 module, and the actual onboard device also uses this interface to transmit data.
b) Intel 750 1.5 TB solid drive supports the 4x PCIE 3.0 interface, the bandwidth can reach 64 Gb/s, which can meet the bandwidth requirement of the Tegra X2 module and can be used as a simulated substitute for FPGA and external satellite subsystems.
c) Compared with data transmission, the bottleneck of onboard fusion processing is processing, and the alternative I/O scheme will not substantially affect the methods and conclusions of this article.

2) Experiment Data: The experiment in this article is carried out based on the data of a high-resolution remote sensing satellite of China. The panchromatic sensor has a spatial resolution of 0.7 m, and the multispectral sensor has a spatial resolution of 2.8 m, which includes four spectral bands: 1) blue, 2) green, 3) red, and 4) near-infrared. In this article, a section of imaging strip data covering Lijiang City, Yunnan Province, is selected to carry out the experiments, and the size of the region of interest is set to 5000 $\times$ 5000 pixels.

B. Registration Accuracy

In the process of fusion processing in this article, sensor correction is first applied to the panchromatic and multispectral data of the ROI, respectively, and then, the fusion is performed.

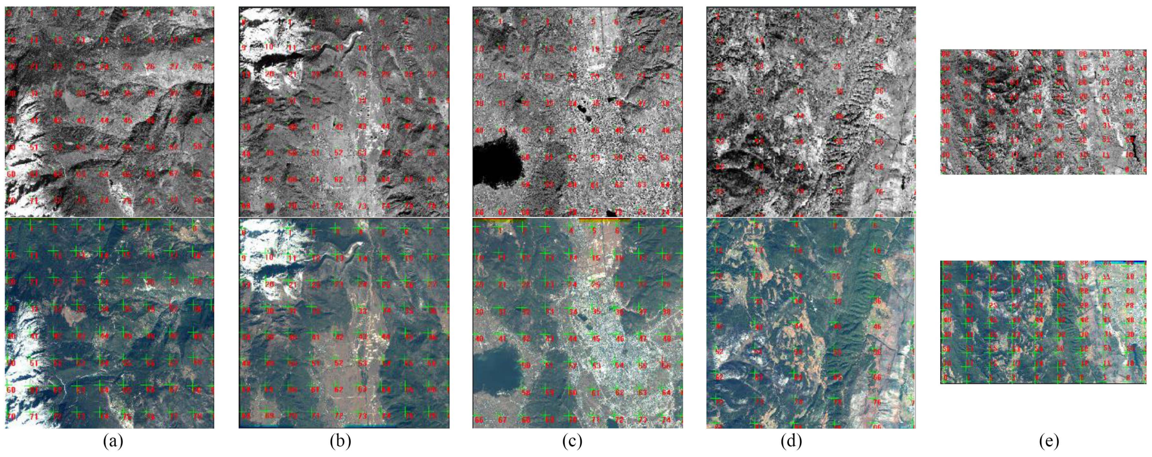


Fig. 15. Relative accuracy of panchromatic-multispectral. (a) Scene 1. (b) Scene 2. (c) Scene 3. (d) Scene 4. (e) Scene 5.

TABLE V
RELATIVE ACCURACY OF PANCHROMATIC-MULTISPECTRAL IMAGES (UNIT: PANCHROMATIC PIXEL)

Scene No.	Avg-X	Avg-Y	RMSE-X	RMSE-Y
1	0.256	-0.461	0.809	0.953
2	0.134	-0.463	0.921	0.992
3	0.444	0.528	0.894	0.999
4	0.106	-0.044	0.748	0.902
5	0.337	0.057	0.862	0.976

Fusion processing can be implemented on embedded hardware by omitting the pixel-based registration process. Taking advantage of the almost simultaneous imaging characteristics of the panchromatic and multispectral images, this article fine-tunes the parameters of the virtual sensors based on the principle of object-space consistency, so as to obtain geometrically consistent panchromatic and multispectral images through the sensor correction and, then, fuse them to get the ROI fusion product. In order to verify the registration accuracy of this method, this section generates panchromatic and multispectral sensor calibration products, respectively, and measures the relative accuracy of the corresponding images through grid matching, as shown in Fig. 15.

Table V lists the relative accuracy experimental results of five standard scenes. It can be found that the method in this article can keep the geometric deviation of panchromatic and multispectral images within one panchromatic pixel through sensor correction, which can meet the fusion processing requirements.

C. ROI Fusion Product

With the data length of 2.4 s (about one standard scene) as an interval, 5 ROI area products (5000 $\times$ 5000 pixels) are continuously generated by the task-driven fusion processing method in this article. The red box in the ROI product intercepts a small area of 500 $\times$ 500 pixels for comparison (with details of 100 $\times$ 100 pixels), as shown in Figs. 16 –20. It can be seen that the application of the ROI fusion processing method based on the principle of object-space consistency in this article can effectively incorporate spatial details while maintaining the spectral characteristics of the original multispectral image. Compared with the original panchromatic and multispectral products, the fusion product contains more information.

It can be seen from Fig. 16 that the spectral characteristics of the fused image are consistent with the multispectral image, in the vegetation scene, the grass texture is clearly visible, the edge outline of the trees is clear enough, and the vehicles on the road are also clearly identifiable.

It can be seen from Fig. 17 that the spectral characteristics of the fused image are consistent with the multispectral image, the boundaries of the bushes on the natural wasteland are clear, and the green color of the pond indicates that the water body is rich in algae.

It can be seen from Fig. 18 that the spectral characteristics of the fused urban image are consistent with the multispectral image, the edges of small objects are clear, and the color reproduction is accurate.

It can be seen from Fig. 19 that the photovoltaic power generation panels on the roofs of residents can be clearly observed in the fused suburban scene image, and the color reproduction of factory buildings and bare soil with large contrast with the surrounding features is accurate.

It can be seen from Fig. 20 that in the fused image, the fields planted with different crops are clearly layered, and the plants are clearly identifiable.

D. Performance

This article performs performance experiments based on a simulated embedded GPU platform. First, the performance of the traditional method based on the standard scene is measured and used as a baseline (Method 1). That is, the original data are first automatically divided into scenes, the size of the panchromatic scene is set to about 29 000 lines (2.4 s), and the size of the standard scene is about 26 000 $\times$ 29 000 pixels; then, sensor correction, panchromatic multispectral fusion, and geometric correction are performed on the standard scene

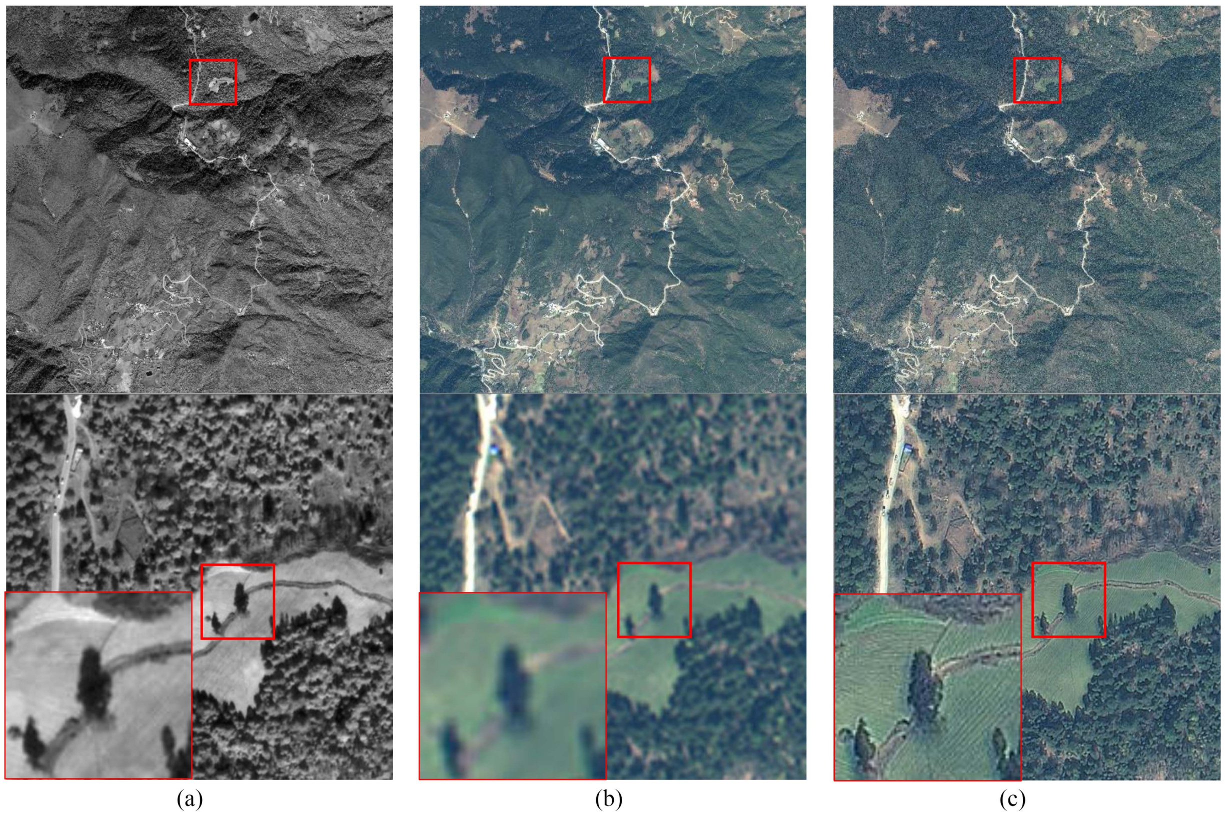


Fig. 16. Textures of vegetations in ROI 1 (Lon:100.26284050, Lat:27.16015811). (a) PAN. (b) MS. (c) FUSION.

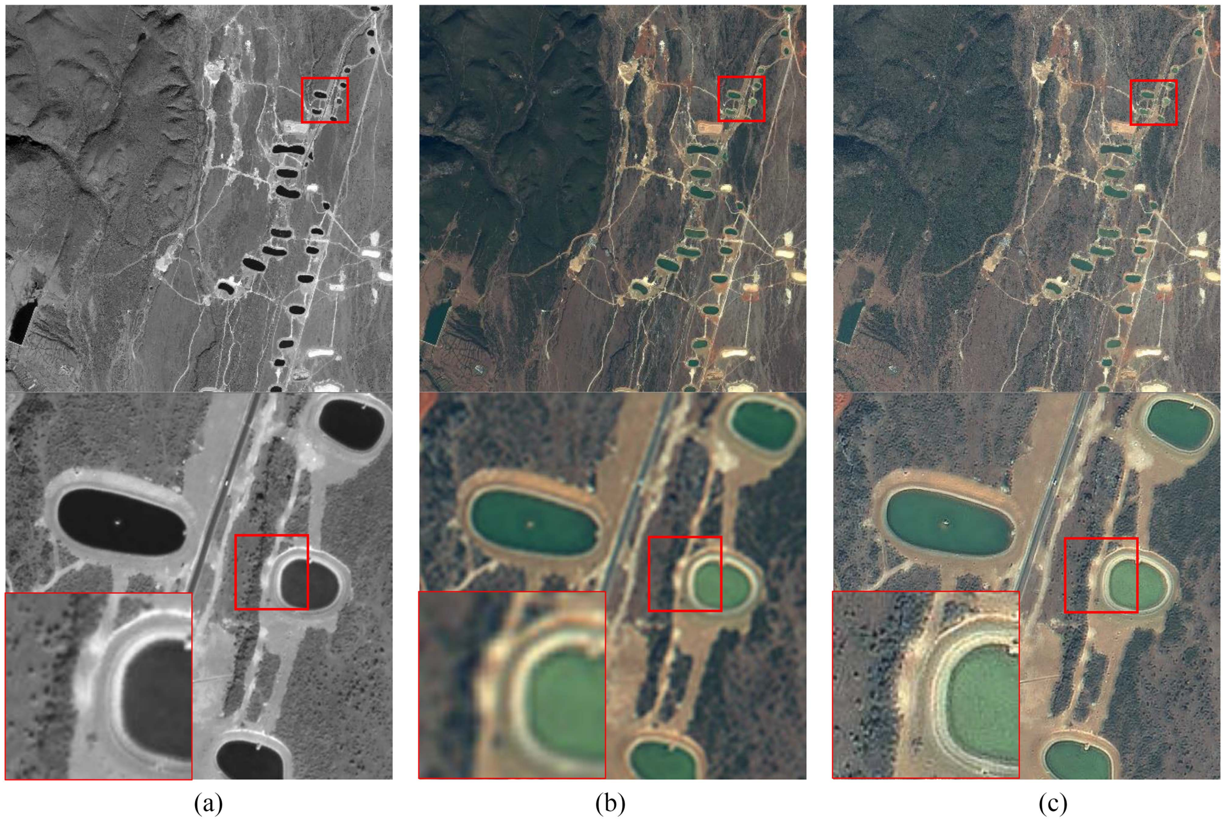


Fig. 17. Color of the waterbody in ROI 2 (Lon:100.24079662, Lat:27.03169744). (a) PAN. (b) MS. (c) FUSION.

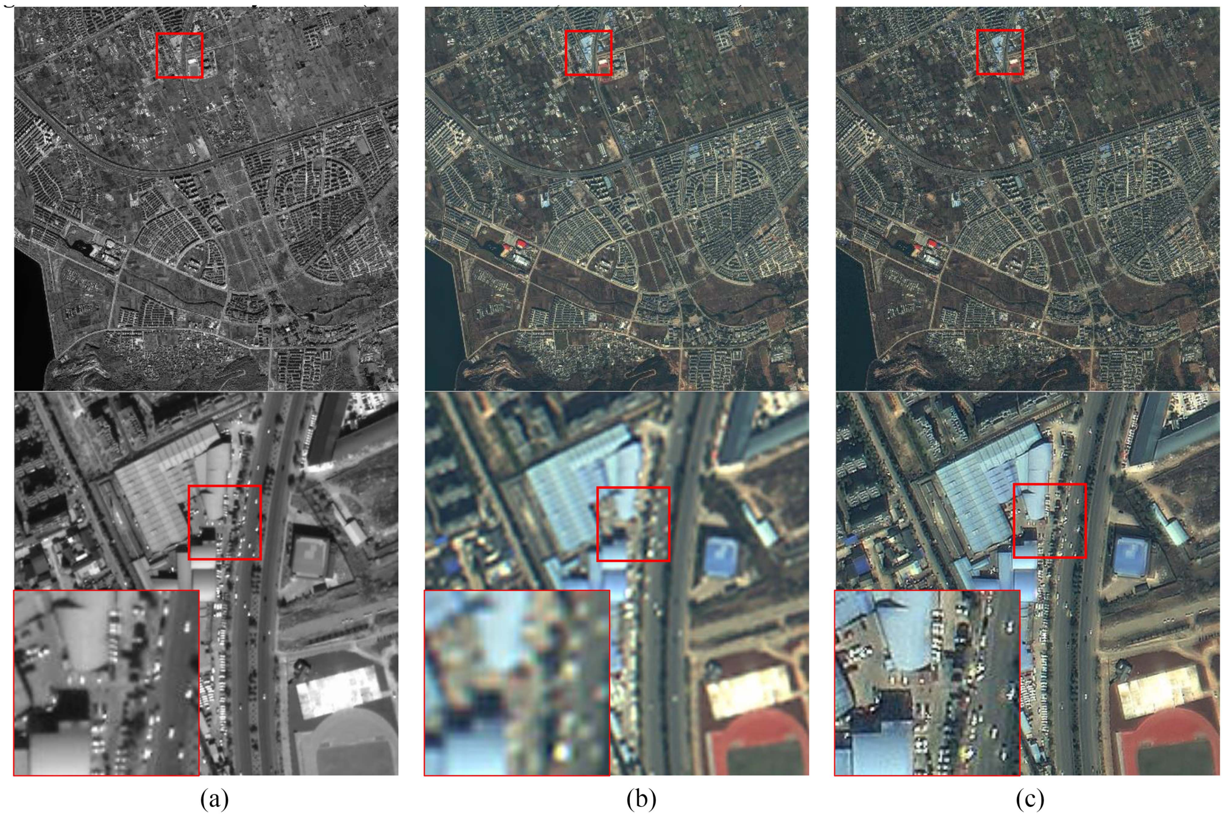


Fig. 18. Edges of small objects in ROI 3 (Lon:100.22883569, Lat:26.83364415). (a) PAN. (b) MS. (c) FUSION.

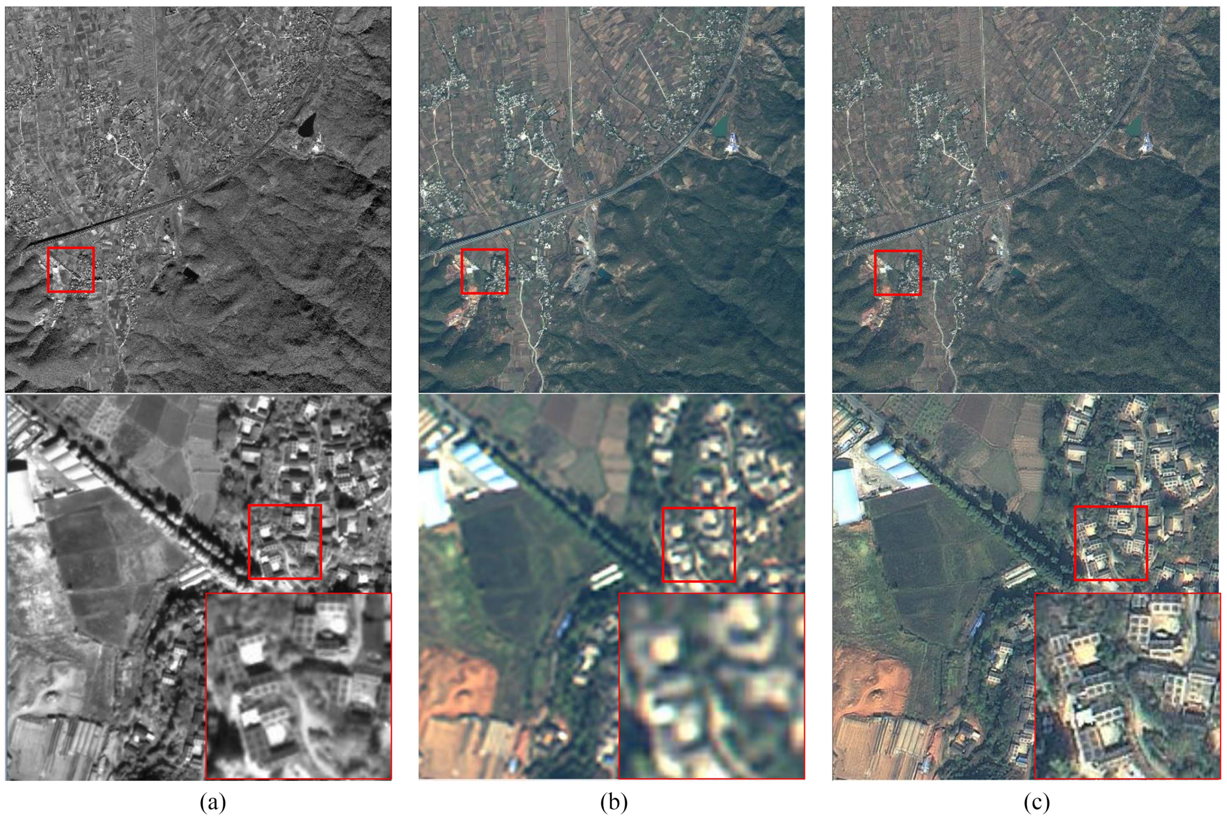


Fig. 19. Roofs of residents in ROI 4 (Lon:100.13476250, Lat:26.82424107). (a) PAN. (b) MS. (c) FUSION.

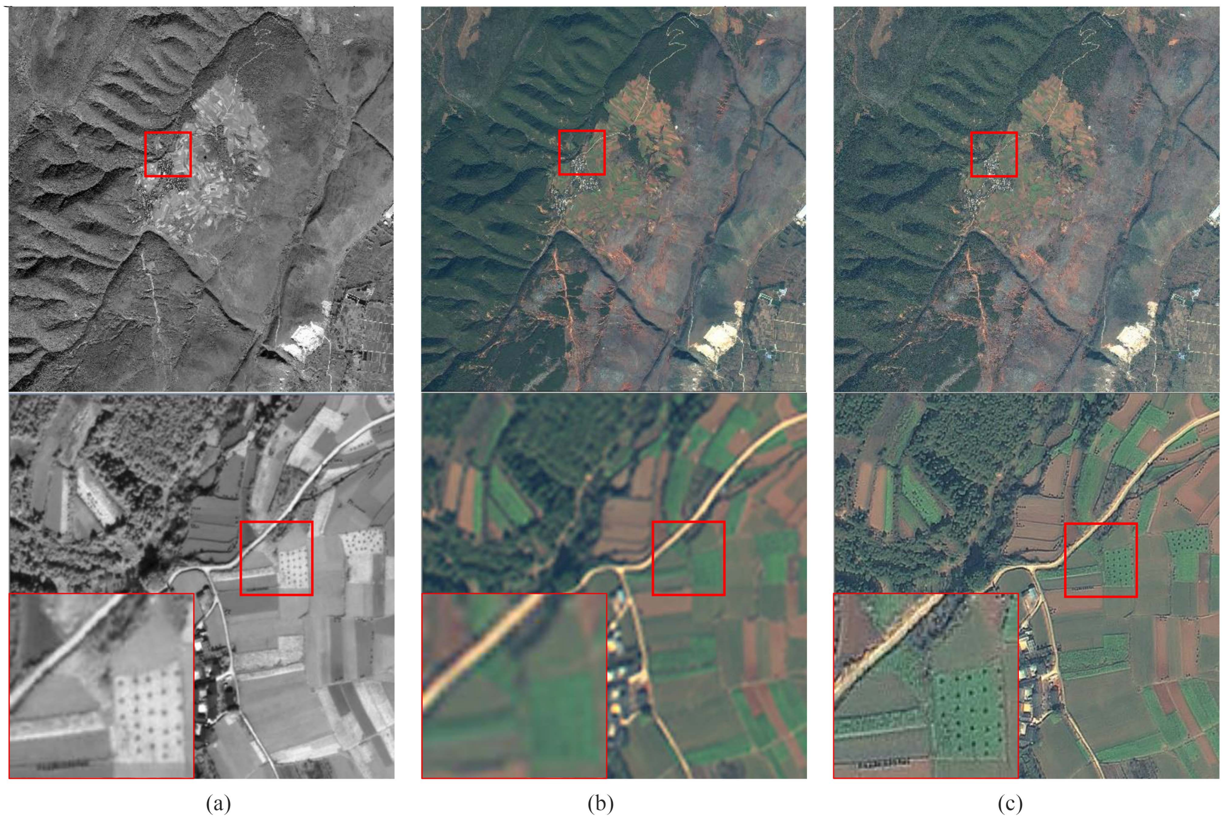


Fig. 20. Layers of plants in ROI 5 (Lon:100.17543804, Lat:26.66730138). (a) PAN. (b) MS. (c) FUSION.

image. The method uses *openmp* for parallel acceleration where appropriate.

Due to the large amount of image data, most methods that first register the entire panchromatic and multispectral images and then fuse them either rely on external storage or require a large amount of memory to store temporary data. Since this article discusses onboard processing in a limited platform, all methods involved in this article are done using only internal memory. And limited by Tegra X2's hardware with only 8 GB of internal memory, the fusion processing method using large internal memory or external memory cannot be realized in this circumstance. Therefore, the parallel progressive fusion algorithm and the registration strategy based on object-space consistency proposed in this article have been already adopted in Method 1.

Furthermore, this article adopts the task-driven onboard fusion processing strategy to improve the baseline method, that is, while the data are flow-in, only the data in the ROI are extracted and processed according to the real-time high-precision geometric locating calculation. Thereby significantly reducing the overall calculation load and improving processing timeliness (Method 2).

On this basis, the algorithm mapping strategy proposed in this article is applied to transform the time-consuming steps into GPU parallelization to achieve a significant improvement in processing performance (Method 3).

TABLE VI
TIME CONSUMPTION OF KERNEL FUNCTION (UNIT: SECOND)

Kernel function	**Method 3**	**Method 3+***	**Method 4***	***S***
resample2D	0.909	0.492	0.466	1.95
resample3D	1.216	0.659	0.302	4.03
fusion	3.202	0.351	0.211	15.18
total	5.327	1.502	0.979	5.44

* Only the optimal value is given in the table because there are too many data.

Finally, in order to further improve the performance of the onboard processing bottleneck algorithm, the CUDA kernel functions involved in Method 3 were deeply optimized and modified: through a large number of experiments, the execution parameters of the kernel functions were deeply optimized (see: Appendix, Method 3+); and according to the ability of Tegra X2, the kernel functions are partially simplified and approximated, and the further CPU/GPU heterogeneous parallel optimization is applied for them (Method 4). As shown in Table VI, the performance of CUDA kernel functions has been improved by 5.44 times, which has brought a significant improvement in the overall performance.

In order to objectively measure the performance improvement effect of different methods, as shown in the following equation,

TABLE VII
PERFORMANCE COMPARISON (UNIT: SECOND)

type	Data analysis	Sensor Correction			Fusion and geocorrection			Sum of Processing	Sum	*S*
		RPCs Calculation	PAN Resampling	MS Resampling	Stitching	Fusion	Geometric Correction			
Method1	2.269	0.351	76.756	19.799	0.026	336.134	225.685	658.751	661.020	1.00
Method2	2.234	0.49	2.372	0.963	0.028	11.270	7.527	22.65	24.884	26.56
Method3	2.311	0.308	1.396	0.377	0.020	3.900	1.463	7.464	9.775	67.62
Method4	1.939	0.379	0.902	0.298	0.022	0.285	0.406	2.292	4.231	156.23

TABLE VIII
TIME CONSUMPTION OF ROIS (UNIT: SECOND)

ROI	Longitude of ROI center	Latitude of ROI center	Data Analysis	Fusion Processing	Data Output	Begin time	End time
0	100.26284050	27.16015811	2.306	2.403	2.485	1.186	8.380
1	100.24079662	27.03169744	2.147	2.172	2.339	3.492	10.719
2	100.22883569	26.83364415	2.208	2.365	2.289	6.418	13.280
3	100.13476250	26.82424107	2.045	2.014	2.374	8.626	15.654
4	100.17543804	26.66730138	1.826	2.108	2.207	10.671	17.861

this article uses the speedup ratio S to measure the degree of performance improvement:

$$S = \frac{M}{T_2} / \frac{M}{T_1} = \frac{T_1}{T_2} \tag{22}$$

where M represents calculation amount, and T_1 and T_2 are the processing time of Method 1 and Method 2 participating in the comparison, respectively.

It can be seen from Table VII that the resampling, fusion, and geometric correction processing in the baseline takes a long time, accounting for the vast majority of the total time consumption, and is the bottleneck of the onboard fusion processing. After applying the task-driven onboard fusion processing strategy proposed in this article, the ROI area is located by instant geographic positioning while the data are flow-in, so that only the data of the target ROI area (5000 $\times$ 5000 pixels) be processed, which can greatly reduce the amount of calculation and significantly shorten the processing time. So that Method 2 is accelerated by 26.56 times compared with the baseline; on this basis, after applying the algorithm mapping strategy of this article to transform the bottleneck algorithms to GPU, Method 3 has obtained a 67.62 times acceleration; furthermore, combining algorithm features and device features, after deeply optimizing the algorithm migrated to GPU, Method 4 has obtained a 156.23 times acceleration.

The processing time of Method 4 includes data analysis t_0(1.939 s), processing t_1 (2.292 s), and data output t_2 (2.411 s), which can basically complete the processing of 5000 $\times$ 5000 pixels ROI from every 2.4 s data in time (roughly equivalent to a standard scene).

From the above results, it can be seen that the time consumption of the steps of the optimized onboard fusion are roughly the same. A parallel pipeline can be built based on the multitasking capability of the operating system carried by the embedded GPU, so as to achieve the purpose of hiding transmission and analysis delays and further improving real-time performance. Considering that there are random fluctuations in the time consumption of algorithm processing and I/O, when building the parallel pipeline, it is necessary to configure sufficient buffer between each step to improve system stability. In this article, the experiment is carried out according to the frequency of 1 ROI in 2.4s, and a total of 5 ROI products are obtained. The actual time consumption records are shown in Table VIII.

It can be seen from Table VIII and Fig. 21 that after constructing the parallel pipeline according to the measured performance, the system can basically complete the real-time fusion processing of 1 ROI in every 2.4 s. However, because the ROI2 area is slightly far away from the ROI1 area, the system is temporarily idle for a short time and some data is backlogged, making the whole processing completion time longer than expected. It is particularly worth noting that in pipeline processing, the impact of similar waiting will gradually accumulate until the data exceeds the buffer and cause the pipeline to crash. Therefore, in the actual industrial system that needs to run for a long time, it is necessary to add auxiliary security strategies. For example, discard ROI tasks that are delayed or too late to process, or leave a certain margin in the settings to ensure the correct operation of the pipeline system.

V. DISCUSSION

In this article, the embedded GPU Terga X2 (TDP:15 W) is adopted as a simulation low-power onboard processing platform, and research is carried out based on this platform. First of all, while the data are flowing in, through real-time positioning, the ROI area can be quickly located and extracted to avoid wasting precious resources due to processing a large number of irrelevant data; on this basis, the fusion process and algorithm are reconstructed, and based on the principle of object-space consistency, the registration of panchromatic and multispectral data is realized in the sensor correction step by adjusting the virtual sensor parameters; furthermore, referring to the computing resource characteristics of the embedded GPU,

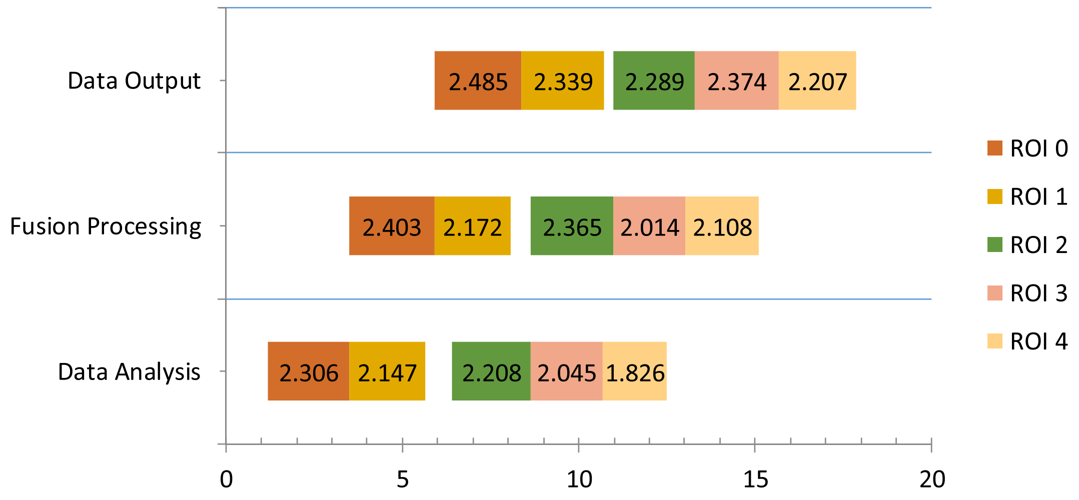
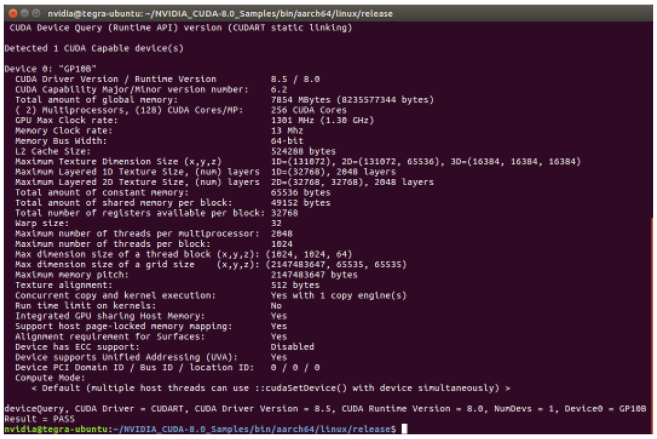


Fig. 21. Timeline of stream computing.

a reasonable algorithm mapping strategy was formulated and deeply optimized to greatly improve the performance, and a pipelined parallel processing was constructed based on this. The experimental results show that the relative accuracy between the panchromatic and multispectral images before fusion is less than one panchromatic pixel, which can meet the requirements of fusion processing; the spectral characteristics of the real-time fusion product are highly consistent with the original image, and the spatial details are rich, which can meet the needs of visual interpretation. In the performance experiment, after reasonable algorithm mapping and deep optimization, compared with the baseline, the performance has been improved by 156.23 times, and real-time processing can be completed at the frequency of 1 ROI (5000 $\times$ 5000 pixels) every 2.4 s.

VI. Conclusion

Constrained by the space environment, onboard processing of remote sensing satellites faces the problems of limited computing power, large amount of data, and high timeliness requirements, which are very difficult and challenging. This article focuses on the onboard real-time fusion processing requirements of high-resolution optical remote sensing satellites and explores a real-time fusion processing approach for the ROI area under the conditions of limited volume and power consumption. The experimental results show that under certain prerequisites, the approach proposed in this article can realize the real-time fusion processing of high-resolution optical satellites and obtain usable fusion results. However, this article mainly focuses on the realization of real-time fusion processing in a limited onboard environment, focusing on efficiency rather than the fusion effect. In subsequent research, it is necessary to focus on the improvement of the fusion effect. In addition, under the framework of the approach in this article, the relative accuracy of the sensor correction results between panchromatic and multispectral is about 0.9 panchromatic pixels in both the x and y directions. Theoretically, it is also possible to improve the geometric consistency by improving the camera calibration accuracy, thereby improving the fusion effect. From the perspective of practical application, the approach in this article is implemented based on embedded GPU. In the future, the adaptation method based on other hardware systems such as DSP, FPGA, and other kinds of embedded devices can be studied according to actual application requirements.


```
nvidia@tegra-ubuntu: ~/NVIDIA_CUDA-8.0_Samples/bin/aarch64/linux/release
 CUDA Device Query (Runtime API) version (CUDART static linking)

Detected 1 CUDA Capable device(s)

Device 0: "GP10B"
  CUDA Driver Version / Runtime Version          8.5 / 8.0
  CUDA Capability Major/Minor version number:    6.2
  Total amount of global memory:                 7854 MBytes (8235577344 bytes)
  ( 2) Multiprocessors, (128) CUDA Cores/MP:     256 CUDA Cores
  GPU Max Clock rate:                            1301 MHz (1.30 GHz)
  Memory Clock rate:                             13 Mhz
  Memory Bus Width:                              64-bit
  L2 Cache Size:                                 524288 bytes
  Maximum Texture Dimension Size (x,y,z)         1D=(131072), 2D=(131072, 65536), 3D=(16384, 16384, 16384)
  Maximum Layered 1D Texture Size, (num) layers  1D=(32768), 2048 layers
  Maximum Layered 2D Texture Size, (num) layers  2D=(32768, 32768), 2048 layers
  Total amount of constant memory:               65536 bytes
  Total amount of shared memory per block:       49152 bytes
  Total number of registers available per block: 32768
  Warp size:                                     32
  Maximum number of threads per multiprocessor:  2048
  Maximum number of threads per block:           1024
  Max dimension size of a thread block (x,y,z): (1024, 1024, 64)
  Max dimension size of a grid size    (x,y,z): (2147483647, 65535, 65535)
  Maximum memory pitch:                          2147483647 bytes
  Texture alignment:                             512 bytes
  Concurrent copy and kernel execution:          Yes with 1 copy engine(s)
  Run time limit on kernels:                     No
  Integrated GPU sharing Host Memory:            Yes
  Support host page-locked memory mapping:       Yes
  Alignment requirement for Surfaces:            Yes
  Device has ECC support:                        Disabled
  Device supports Unified Addressing (UVA):      Yes
  Device PCI Domain ID / Bus ID / location ID:   0 / 0 / 0
  Compute Mode:
     < Default (multiple host threads can use ::cudaSetDevice() with device simultaneously) >

deviceQuery, CUDA Driver = CUDART, CUDA Driver Version = 8.5, CUDA Runtime Version = 8.0, NumDevs = 1, Device0 = GP10B
Result = PASS
nvidia@tegra-ubuntu:~/NVIDIA_CUDA-8.0_Samples/bin/aarch64/linux/release$ 
```

Fig. 22. Parameters of Tegra X2.

Appendix

Since there are many factors that affect the performance of CUDA kernel functions, it is necessary to understand the influence of core parameter settings such as thread block size, occupancy, and number of registers used during execution on performance and then carry out targeted optimization.

Using the *deviceQuery* program that comes with CUDA on Tegra X2, the main performance parameters of Tegra X2 are shown in Fig. 22. It can be seen that the thread package (Warp) size it supports is 32, and the maximum thread block size is 1024. Therefore, first, adjust the thread block parameters (T_x, T_y) of the onboard algorithm and analyze the theoretical and actual occupancy and time consumption of kernel functions in different

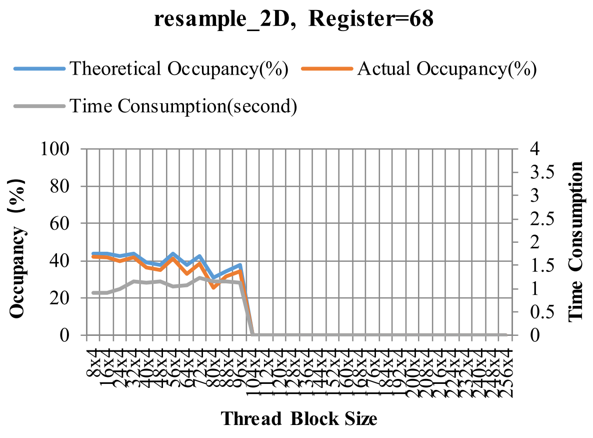


Fig. 23. Resample_2D (× 4).

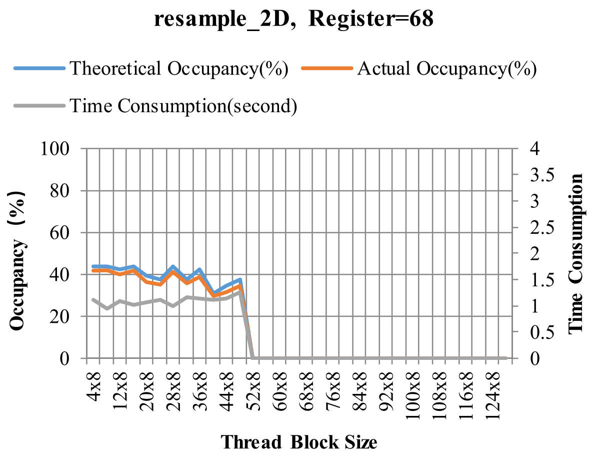


Fig. 24. Resample_2D (× 8).

situations. Take T_y= 4, T_x take 8, 16, 24,…,256 in turn; then, take T_y= 8, T_x take 4, 8, 12,…,128 in turn. After changing the settings each time, recompile the program and execute it, use the *nvprof* tool to measure the theoretical occupancy, actual occupancy, time consumption, register usage, and other related dynamic parameter values of the kernel functions, and then choose the best setting with the best performance. The Appendix shows the tuning process of Methods 3 to Method 4 in the performance experiment of this article.

Step1: The register requirements of the kernel functions (resample_2D, resample_3D, and fusion) are (68, 68, and 40). The statistics of the execution results of the three kernel functions under different settings are as follows.

1) *resample_2D:* Take T_y= 4, T_x take 8, 16, 24,…,256 in turn, the statistics of the execution results of the kernel function resample_2D is shown in Fig. 23.

Take T_y= 8, T_x take 4, 8, 12,…,128 in turn, the statistics of the execution results of the kernel function resample_2D is shown in Fig. 24.

2) *resample_3D:* Take T_y= 4, T_x take 8, 16, 24,…,256 in turn, the statistics of the execution results of the kernel function resample_3D is shown in Fig. 25.

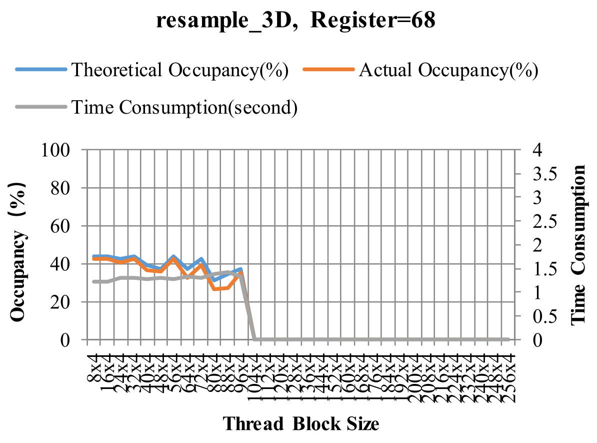


Fig. 25. Resample_3D (× 4).

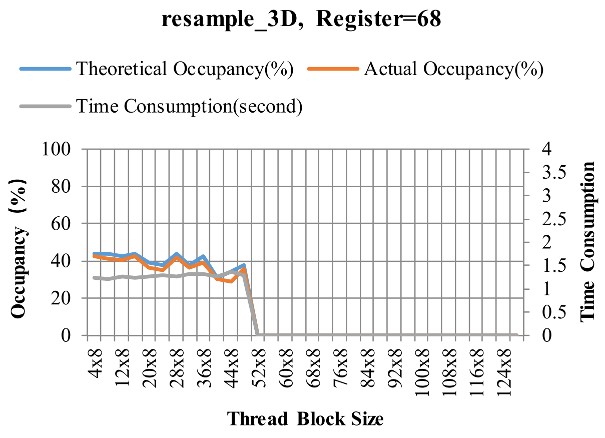


Fig. 26. Resample_3D (× 8).

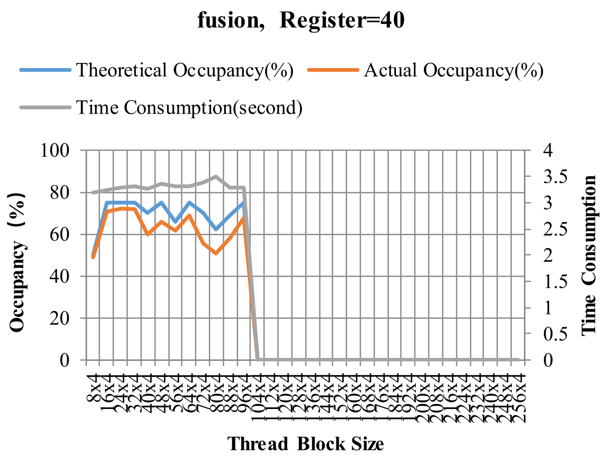


Fig. 27. Fusion (× 4).

Take T_y= 8, T_x take 4, 8, 12,…,128 in turn, the statistics of the execution results of the kernel function resample_3D is shown in Fig. 26.

3) *fusion:* Take T_y= 4, T_x take 8, 16, 24,…,256 in turn, the statistics of the execution results of the kernel function fusion is shown in Fig. 27.

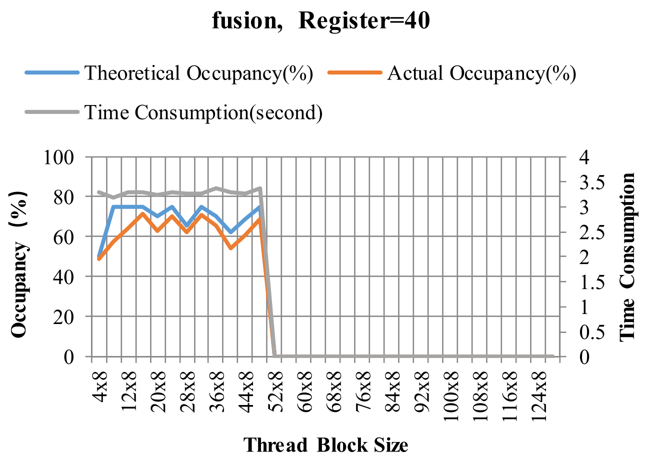


Fig. 28. Fusion (× 8).

TABLE IX
TIME CONSUMPTION OF KERNEL FUNCTION

Kernel function	Time Consumption (s)	Call Times	Processing time (ms)
resample_2D	0.909	840	1.082
resample_3D	1.216	400	3.040
fusion	3.202	400	8.005

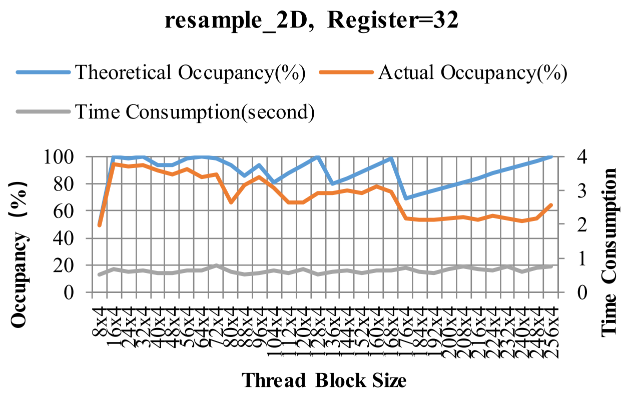


Fig. 29. Resample_2D (× 4).

Take $T_y = 8$, T_x take 4, 8, 12, …,128 in turn, the statistics of the execution results of the kernel function fusion is shown in Fig. 28.

Table IX shows the time consumption of kernel function.

Step2: After applying kernel function optimization, CUDA core occupancy optimization, and kernel function storage access optimization, the register requirements of the kernel functions (resample_2D, resample_3D, and fusion) reduced from (68, 68, and 40) to (32, 64, and 32). The statistics of the execution results of the three kernel functions under different settings are as follows.

1) *resample_2D:* Take $T_y = 4$, T_x take 8, 16, 24, …,256 in turn, the statistics of the execution results of the kernel function resample_2D is shown in Fig. 29.

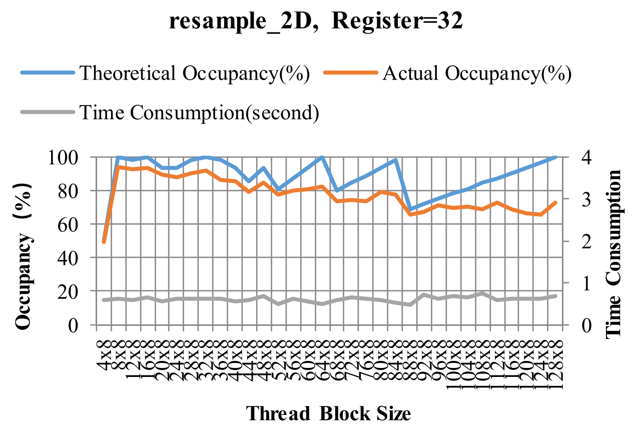


Fig. 30. Resample_2D (× 8).

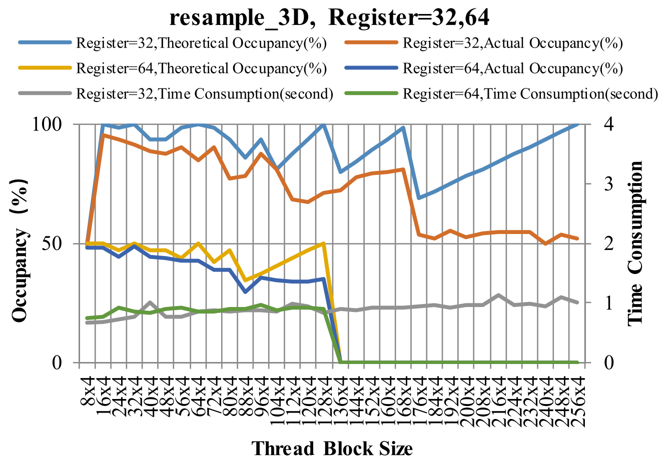


Fig. 31. Resample_3D (× 4).

Take $T_y = 8$, T_x take 4, 8, 12, …,128 in turn, the statistics of the execution results of the kernel function resample_2D is shown in Fig. 30.

2) *resample_3D:* Take $T_y = 4$, T_x take 8, 16, 24, …,256 in turn,the statistics of the execution results of the kernel function resample_3D is shown in Fig. 31.

Take $T_y = 8$, T_x take 4, 8, 12, …,128 in turn, the statistics of the execution results of the kernel function resample_3D is shown in Fig. 32.

3) *fusion:* Take $T_y = 4$, T_x take 8, 16, 24, …,256 in turn, the statistics of the execution results of the kernel function fusion is shown in Fig. 33.

Take $T_y = 8$, T_x take 4, 8, 12, …,128 in turn, the statistics of the execution results of the kernel function fusion is shown in Fig. 34 .

As shown in Table X, compared with Method 3, the performance of the CUDA kernel function in Method 3+ has been improved by 3.55 times.

Step3: After the kernel functions are partially simplified and approximated, the register requirements of the kernel functions (resample_2D, resample_3D, and fusion) are further reduced from (32, 64, and 32) to (30, 32, and 26). The register requirements of all kernel functions are lower than the threshold of 32

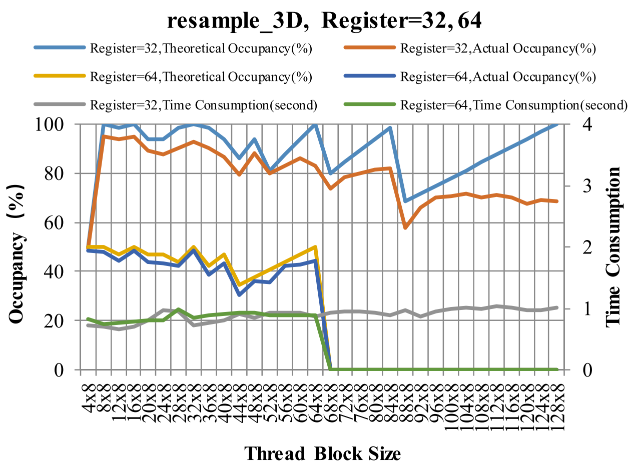


Fig. 32. Resample_3D (× 8).

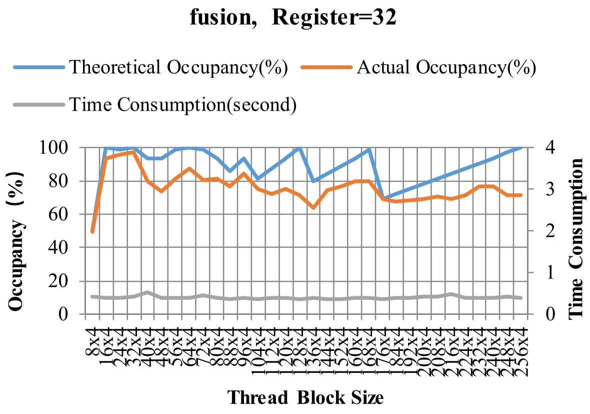


Fig. 33. Fusion (× 4).

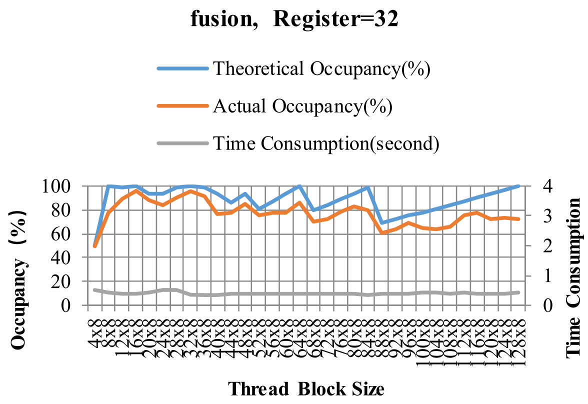


Fig. 34. Fusion (× 8).

TABLE X
TIME CONSUMPTION OF KERNEL FUNCTION

Kernel function	Method 3 (s)	Method 3+(s)*
resample_2D	0.909	0.492
resample_3D	1.216	0.659
fusion	3.202	0.351
total	5.327	1.502

* Only the optimal value is given in the table because there are too many data.

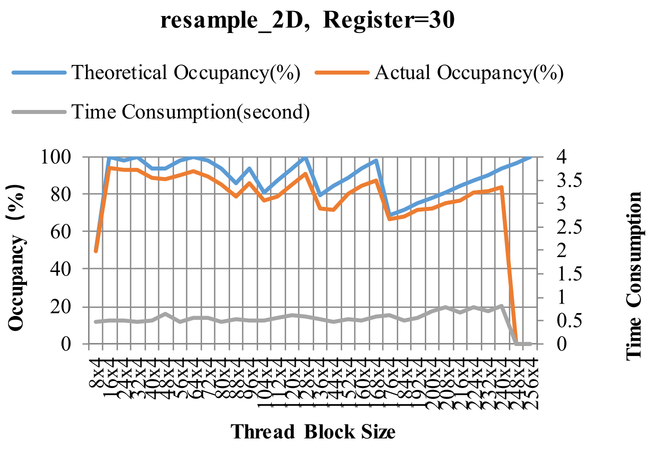


Fig. 35. Resample_2D (× 4).

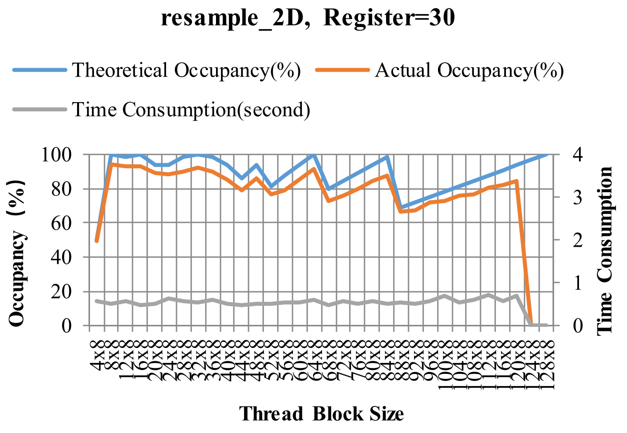


Fig. 36. Resample_2D (× 8).

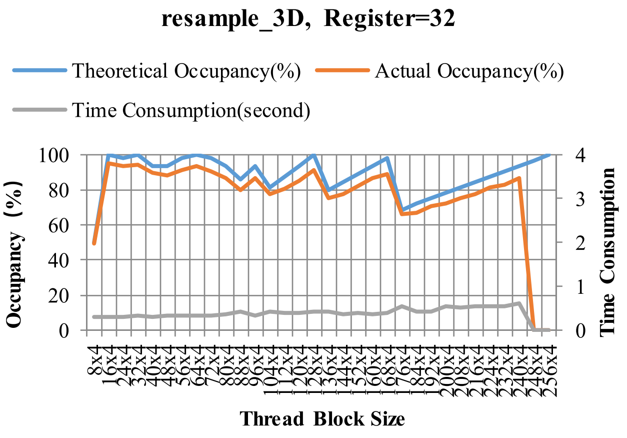


Fig. 37. Resample_3D (× 4).

of Tegra X2, which means that the performance of all kernel functions is no longer limited by the number of registers of hardware devices. The statistics of the execution results of the three kernel functions under different settings are as follows.

1) *resample_2D:* Take $T_y = 4$, T_x take 8, 16, 24, …,256 in turn, the statistics of the execution results of the kernel function resample_2D is shown in Fig. 35.

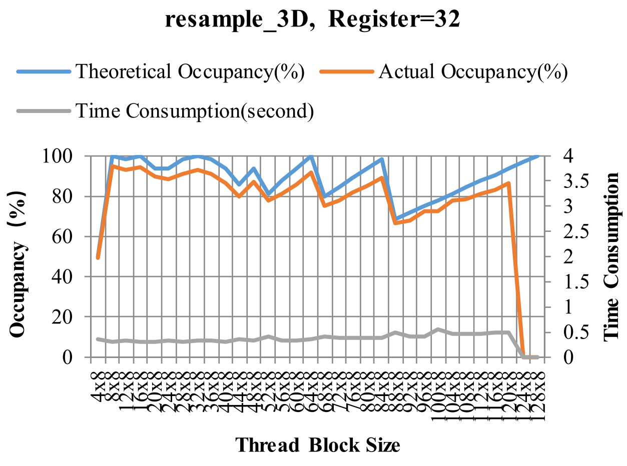


Fig. 38. Resample_3D (× 8).

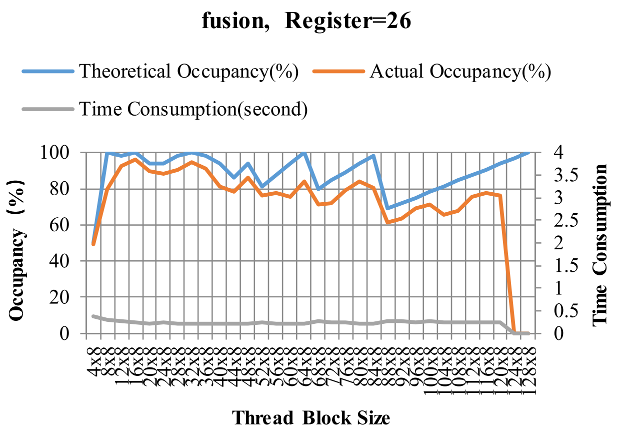


Fig. 40. Fusion (× 8).

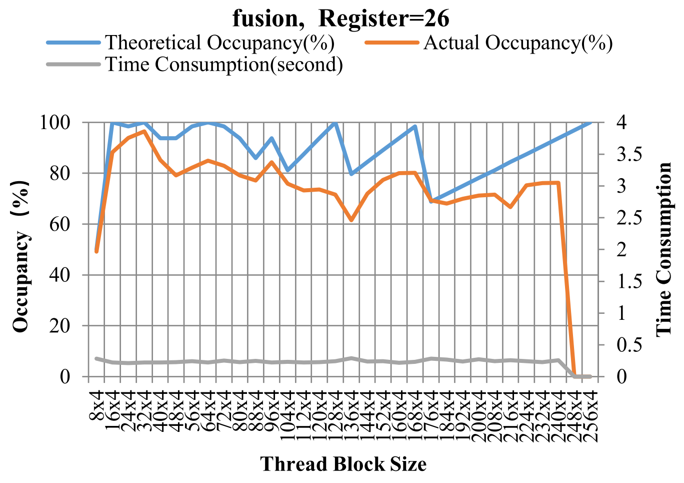


Fig. 39. Fusion (× 4).

Take T_y= 8, T_x take 4, 8, 12,…,128 in turn, the statistics of the execution results of the kernel function resample_2D is shown in Fig. 36.

2) *resample_3D:* Take T_y= 4, T_x take 8, 16, 24,…,256 in turn, the statistics of the execution results of the kernel function resample_3D is shown in Fig. 37.

Take T_y= 8, T_x take 4, 8, 12,…,128 in turn, the statistics of the execution results of the kernel function resample_3D is shown in Fig. 38.

3) *fusion:* Take T_y= 4, T_x take 8, 16, 24,…,256 in turn, the statistics of the execution results of the kernel function fusion is shown in Fig. 39.

Take T_y= 8, T_x take 4, 8, 12,…,128 in turn, the statistics of the execution results of the kernel function fusion is shown in Fig. 40.

As shown in Table XI, compared with Method 3, the performance of the CUDA kernel function in Method 4 has been improved by 5.44 times, which has brought a significant improvement in the overall performance of Method 4.

TABLE XI
TIME CONSUMPTION OF KERNEL FUNCTION

Kernel Function	Method 3 (s)	Method 3+ (s)*	Method 4 (s)*
resample_2D	0.909	0.492	0.466
resample_3D	1.216	0.659	0.302
fusion	3.202	0.351	0.211
total	5.327	1.502	0.979

* Only the optimal value is given in the table because there are too many data.

ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers and members of the editorial team for their comments and suggestions.

REFERENCES

[1] G. Dial et al., "IKONOS satellite, imagery, and products," *Remote Sens. Environ.*, vol. 88, no. 1, pp. 23–36, 2003.
[2] D. Li, "China's first civilian three-line-array stereo mapping satellite: ZY-3," *Acta Geodaetica et Cartographica Sinica*, vol. 41, no. 3, pp. 317–322, 2012.
[3] J. Y. Yang. *Study on Parallel Processing Technologies of Photogrammetry Data Based on GPU*. Zhengzhou, China: Inf. Eng. Univ., 2011.
[4] D. Li et al., *On Construction of China's Space Information Network*. Wuhan, China: Geomatics Inf. Sci. Wuhan Univ., vol. 40, no. 6, pp. 711–715, 2015.
[5] D. Li, "Towards geo-spatial information science in big data era," *Acta Geodaetica et Cartographica Sinica*, vol. 45, no. 4, pp. 379–374, 2016.
[6] Z. Bing, "Intelligent remote sensing satellite system," *J. Remote Sens.*, vol. 15, no. 3, pp. 415–431, 2011.
[7] W. Mi et al., "Stream-computing based high accuracy on-board real-time cloud detection for high resolution optical satellite imagery," *Acta Geodaetica Cartographica Sinica*, vol. 47, no. 6, pp. 760–769, 2018.
[8] G. Xie et al., "Near real-time automatic sub-pixel registration of panchromatic and multispectral images for pan-sharpening," *Remote Sens.*, vol. 13, no. 18, 2021, Art. no. 3674.
[9] G. Xie et al., "On-board GCPS matching with improved triplet loss function," *ISPRS Ann. Photogrammetry, Remote Sens., Spatial Inf. Sci.*, vol. 2, pp. 105–112, 2020.
[10] M. Wang, G. Xie, Z. Zhang, Y. Wang, S. Xiang, and Y. Pi, "Smoothing filter-based panchromatic spectral decomposition for multispectral and hyperspectral image pansharpening," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 15, no. 4, pp. 3612–3625, Apr. 2022.
[11] S. Xiang et al., "Dual-task semantic change detection for remote sensing images using the generative change field module," *Remote Sens.*, vol. 13, no. 16, 2021, Art. no. 3336.

[12] Z. Zhang, H. Zheng, J. Cao, X. Feng, and G. Xie, "FRS-Net: An efficient ship detection network for thin-cloud and fog-covered high-resolution optical satellite imagery," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 15, no. 12, pp. 2326–2340, Dec. 2022.
[13] Z. Zhang et al., "MKANet: An efficient network with sobel boundary loss for land-cover classification of satellite remote sensing imagery," *Remote Sens.*, vol. 14, no. 18, 2022, Art. no. 4514.
[14] B. Zhang et al., "A high reliability on-board parallel system based on multiple DSPs," in *Proc. Int. Conf. Mechatron. Sci.*, 2013, pp. 1984–1988.
[15] K. Hänsler et al., "TID and SEE performance of a commercial 0.13μm CMOS technology," in *Proc. Radiat. Effects Compon. Syst.*, vol. 536, pp. 119–125.
[16] D. N. Nguyen and F. Irom, "Total ionizing dose (TID) tests on non-volatile memories: Flash and MRAM," in *Proc. Radiat. Effects Data Workshop*, 2007, pp. 194–198.
[17] L. Ding et al., "Analysis of TID failure modes in SRAM-based FPGA under gamma-ray and focused synchrotron X-ray irradiation," *IEEE Trans. Nucl. Sci.*, vol. 61, no. 4, pp. 1777–1784, Aug. 2014.
[18] D. Yigang, "Single event effects in space radiation environment," *Spacecraft Environ. Eng.*, vol. 24, no. 5, pp. 283–290, 2007.
[19] M. Tafazoli, "A study of on-orbit spacecraft failures," *Acta Astronautica*, vol. 64, no. 2, pp. 195–205, 2009.
[20] J. Plante and H. Shaw, "Evaluation of 3D plus packaging test structures for NASA Goddard Space Flight Center," in *Proc. Eur. Space Compon. Conf.*, 2002, pp. 213–219.
[21] Y. Wang et al., "Study on the deep submicron radiation hardened SRAM circuit for RHBD technologies," *Bandaoti Jishu (Semicond. Technol.)*, vol. 37, no. 1, pp. 18–23, 2012.
[22] A. D. Franklin et al., "Scalable and fully self-aligned n-type carbon nanotube transistors with gate-all-around," in *Proc. Electron Devices Meeting*, 2012, pp. 4–5.
[23] H. Nan, W. Wang, and K. Choi, "Circuit design for carbon nanotube field effect transistors," in *Proc. SoC Des. Conf.*, 2013, pp. 351–354.
[24] A. D. S. Curiel et al., "First results from the disaster monitoring constellation (DMC)," *Acta Astronautica*, vol. 56, no. 1, pp. 261–271, 2005.
[25] S. Yuhaniz, T. Vladimirova, and M. Sweeting, "Embedded intelligent imaging on-board small satellites," in *Proc. Asia-Pacific Conf. Adv. Comput. Syst. Architecture*, 2005, pp. 90–103.
[26] R. J. Norman et al., "Radio occultation measurements from the Australian microsatellite FedSat," *IEEE Trans. Geosci. Remote Sens.*, vol. 50, no. 11, pp. 4832–4839, Nov. 2012.
[27] B. Zhukov et al., "Spaceborne detection and characterization of fires during the bi-spectral infrared detection (BIRD) experimental small satellite mission (2001–2004)," *Remote Sens. Environ.*, vol. 100, no. 1, pp. 29–51, 2006.
[28] W. Halle and W. Skrbek, "Thematic data processing on board the satellite BIRD," in *Proc. Soc. Photographic Instrum. Engineers*, 2000, vol. 4540, pp. 412–419.
[29] S. J. Visser and A. S. Dawood, "Real-time natural disasters detection and monitoring from smart earth observation satellite," *J. Aerosp. Eng.*, vol. 17, no. 1, pp. 10–19, 2004.
[30] D. C. Utley, "NEMO satellite sensor imaging payload," *Proc. SPIE*, vol. 3437, pp. 29–40, 1998.
[31] T. Bretschneider et al., "X-Sat mission progress," in *Proc. Small Satell. Earth Observ., Special Issue Int. Acad. Astronaut.*, 2005, pp. 145–152.
[32] M. Arnaud et al., "The Pleiades optical high resolution program," in *Proc. 57th Int. Astronaut. Congr./Int. Astronaut. Federation/Indian Accounting Assoc. Valencia: Int. Astronaut. Congr.*, 2006, Art. no. IAC-06-B1.
[33] J. Huang and G. Zhou, "On-board detection and matching of feature points," *Remote Sens.*, vol. 9, no. 6, 2017, Art. no. 601.
[34] X. Xuemin, *Research and Implementation of JPEG2000 Satellite Image Compression*. Changsha, China: Nat. Univ. Defense Technol., 2007.
[35] M. Wang et al., "Embedded GPU implementation of sensor correction for on-board real-time stream computing of high-resolution optical satellite imagery," *J. Real-Time Image Process.*, vol. 15, no. 6, pp. 565–581, 2018.
[36] M. Hsueh and C. I. Chang, *Field Programmable Gate Arrays (FPGA) for Pixel Purity Index Using Blocks of Skewers for Endmember Extraction in Hyperspectral Imagery*. Newbury Park, CA, USA: Sage. 2008.
[37] E. El-Araby et al., "Reconfigurable processing for satellite on-board automatic cloud cover assessment," *J. Real-Time Image Process.*, vol. 4, no. 3, pp. 245–259, 2009.
[38] C. González et al., "FPGA implementation of the pixel purity index algorithm for remotely sensed hyperspectral image analysis," *EURASIP J. Adv. Signal Process.*, vol. 2010, no. 1, pp. 1–13, 2010.
[39] Z. Zhang et al., "Expandable on-board real-time edge computing architecture for Luojia3 intelligent remote sensing satellite," *Remote Sens.*, vol. 14, no. 15, 2022, Art. no. 3596.
[40] H. Bock, "Efficient methods for determining precise orbits of low earth orbiters using the global positioning system," Ph.D. dissertation, Astronomical Inst., Univ. Berne, Bern, Switzerland, vol. 65, 2003.
[41] M. Wang et al., "On-orbit geometric calibration model and its applications for high-resolution optical satellite imagery," *Remote Sens.*, vol. 6, no. 5, pp. 4391–4408, 2014.
[42] W. Xu, J. Gong, and M. Wang, "Development, application, and prospects for Chinese land observation satellites," *Geo-Spatial Inf. Sci.*, vol. 17, no. 2, pp. 102–109, 2014.
[43] J. Grodecki and G. Dial, "Block adjustment of high-resolution satellite images described by rational polynomials," *Photogrammetric Eng. Remote Sens.*, vol. 69, no. 1, pp. 59–68, 2003.
[44] C. S. Fraser and H. B. Hanley, "Bias-compensated RPCs for sensor orientation of high-resolution satellite imagery," *Photogrammetric Eng. Remote Sens.*, vol. 71, no. 8, pp. 909–915, 2005.
[45] C. S. Fraser, G. Dial, and J. Grodecki, "Sensor orientation via RPCs," *Int. Soc. Photogrammetry Remote Sens. J. Photogrammetry Remote Sens.*, vol. 60, no. 3, pp. 182–194, 2006.
[46] C. V. Tao and Y. Hu, "A comprehensive study of the rational function model for photogrammetric processing," *Photogrammetric Eng. Remote Sens.*, vol. 67, no. 12, pp. 1347–1358, 2001.
[47] F. Hu *Research on Inner FOV Stitching Theories and Algorithms for Sub-Images of Three Non-Collinear TDI CCD Chips*. Wuhan, China: Wuhan Univ., 2010, pp. 71–77.
[48] J. Pan et al., "Parallel band-to-band registration for HJ-1A1B CCD images using openMP," in *Proc. Int. Symp. Image Data Fusion*, 2011, pp. 1–4.
[49] K. Jacobsen, "Calibration of imaging satellite sensors," *Int. Arch. Photogrammetry Remote Sens.*, vol. 36, pp. 1–7, 2006.
[50] X. Tang et al., "Inner FoV stitching of spaceborne TDI CCD images based on sensor geometry and projection plane in object space," *Remote Sens.*, vol. 6, no. 7, pp. 6386–6406, 2014.
[51] M. Wang et al., "Correction of ZY-3 image distortion caused by satellite jitter via virtual steady reimaging using attitude data," *ISPRS J. Photogrammetry Remote Sens.*, vol. 119, pp. 108–123, 2016.
[52] J. G Liu, "Smoothing filter-based intensity modulation: A spectral preserve image fusion technique for improving spatial details," *Int. J. Remote Sens.*, vol. 21, no. 18, pp. 3461–3472, 2000.
[53] B. Han and Y. Zhao, "An improved smoothing filter-based intensity modulation algorithm for hyperspectral image fusion," *Remote Sens. Inf.*, vol. 27, no. 5, pp. 44–47, 2012.
[54] C. Elachi and P. D. Zimmerman, "Introduction to the physics and techniques of remote sensing," *Phys. Today*, vol. 41, p. 126, 1988.
[55] L. J. Guo, M. J. MOORE, and J. D. Haigh, "Simulated reflectance technique for ATM image enhancement," *Int. J. Remote Sens.*, vol. 18, no. 2. pp. 243–254, 1997.
[56] E. M. Mikhail, J. S. Bethel, and J. C. McGlone *Introduction to Modern Photogrammetry*. Hoboken, NJ, USA: Wiley, 2001.

Zhiqi Zhang received the B.Sc. degree in geographic information system from Huazhong Agricultural University, Wuhan, China, in 2006, the B.Eng. degree in computer science and technology from the Huazhong University of Science and Technology, Wuhan, in 2006, and the M.Eng. degree in computer technology and the Ph.D. degree in photogrammetry and remote sensing from Wuhan University, Wuhan, in 2015 and 2018, respectively.

He is currently an Associate Professor with the School of Computer Science, Hubei University of Technology, Wuhan. His research interests include system architecture, algorithm optimization, AI, and high-performance processing of remote sensing.

Lu Wei received the B.Eng. degree in software engineering from the Wuhan University of Technology, Wuhan, China, in 2006, and the M.Eng. degree in computer technology from Wuhan University, Wuhan, in 2019.

She was a Senior Engineer with Huawei Technologies Corporation and is currently an Associate Professor with the School of Information Science and Engineering, Wuchang Shouyi University, Wuhan. Her research interests include image radiance correction, multisensor image fusion, image quality assessment, and intelligent image processing.

Shao Xiang received the B.S. degree in automation engineering from Hefei University, Hefei, China, in 2017, and the M.S. degree in automation from the College of Electrical and Information Engineering, Hunan University, Changsha, China, in 2020. He is currently working toward the Ph.D. degree in photogrammetry and remote sensing in photogrammetry and remote sensing with the State Key Laboratory of Information Engineering in Surveying, Mapping and Remote Sensing, Wuhan University, Wuhan, China.

His research interests include change detection, image compression, image fusion, object detection, and semantic segmentation of remote sensing.

Guangqi Xie received the M.Sc. degree in remote sensing of resources and environment from the China University of Geosciences, Wuhan, China, in 2018, and the Ph.D. degree in photogrammetry and remote sensing from the State Key Laboratory of Information Engineering in Surveying, Mapping, and Remote Sensing, Wuhan University, Wuhan, in 2021.

He is currently a Lecturer with the School of Computer Science, Hubei University of Technology, Wuhan. His research interests include image matching and registration, pansharpening, and image superresolution.

Chuang Liu received the B.Sc. degree in computer science and technology from the Wuhan University of Engineering Science, Wuhan, China, in 2022. He is currently working toward the M.Eng. degree with the School of Computer Science, Hubei University of Technology, Wuhan.

His research interests include intelligent remote sensing image processing, image fusion, and deep learning.

Mingyuan Xu is currently working toward the B.Eng. degree with the School of Computer Science, Hubei University of Technology, Wuhan, China.

His research interests include digital image processing and deep learning.