cuFSDAF: An Enhanced Flexible Spatiotemporal Data Fusion Algorithm Parallelized Using Graphics Processing Units

Huan Gao, Xiaolin Zhu, Qingfeng Guan, Xue Yang, Yao Yao, Wen Zeng, and Xuantong Peng

***Abstract*—Spatiotemporal data fusion is a cost-effective way to produce remote sensing images with high spatial and temporal resolutions using multisource images. Using spectral unmixing analysis and spatial interpolation, the flexible spatiotemporal data fusion (FSDAF) algorithm is suitable for heterogeneous landscapes and capable of capturing abrupt land-cover changes. However, the extensive computational complexity of FSDAF prevents its use in large-scale applications and mass production. Besides, the domain decomposition strategy of FSDAF causes accuracy loss at the edges of subdomains due to the insufficient consideration of edge effects. In this study, an enhanced FSDAF (cuFSDAF) is proposed to address these problems, and includes three main improvements. First, the TPS interpolator is replaced by an accelerated inverse distance weighted (IDW) interpolator to reduce computational complexity. Second, the algorithm is parallelized based on the compute unified device architecture (CUDA), a widely used parallel computing framework for graphics processing units (GPUs). Third, an adaptive domain decomposition (ADD) method is proposed to improve the fusion accuracy at the edges of subdomains and to enable GPUs with varying computing capacities to deal with datasets of any size. Experiments showed while obtaining similar accuracies to FSDAF and an up-to-date deep-learning-based method, cuFSDAF reduced the computing time significantly and achieved speed-ups of 140.3–182.2 over the original FSDAF program. cuFSDAF is capable of efficiently producing fused images with both high spatial and temporal resolutions to support applications for large-scale and long-term land surface dynamics. Source code and test data available at https://github.com/HPSCIL/cuFSDAF.**

***Index Terms*—Compute unified device architecture (CUDA), multisource satellite images, parallel computing, spatiotemporal data fusion.**

I. Introduction

THE dense time series of satellite images with high spatial resolutions are critical for monitoring land surface dynamics in heterogeneous landscapes. In recent years, satellites with advanced sensors, such as the microsatellites by Planet Labs, WorldView-4, and GF-2 [1], can acquire images with high spatial and temporal resolutions to compose dense time series. However, the high cost of data acquisition from these sensors limits their applications for large-scale land surface dynamics. Furthermore, these sensors are unable to trace long-term historical dynamics. Compared with advanced satellite sensors, long-running satellite sensors can provide long-term, large-scale, and free-of-charge satellite imagery from the past several decades, such as Landsat and MODIS. However, these historical images have either lower spatial resolutions or lower temporal resolutions, limited by hardware technologies [2] and atmospheric conditions [3]. For instance, Landsat provides images with spatial resolutions ranging from 15 to 60 m and a revisit cycle of 16 days. MODIS provides images with spatial resolutions of 250 m–1 km, and the revisit cycle is 1–2 days. Given that these satellite images do not meet the requirements of long-term and large-scale applications of land surface dynamics, spatiotemporal data fusion provides a feasible method for the production of remote sensing images with both high spatial and temporal resolutions.

Spatiotemporal data fusion algorithms combine the spatial information from high spatial resolution images with the temporal information from high temporal resolution images to generate images with both high spatial and temporal resolutions. Existing spatiotemporal data fusion algorithms can be divided into five categories, i.e., unmixing-based, weight function-based, Bayesian-based, learning-based, and hybrid fusion [4]. Unmixing-based algorithms assume that each mixed pixel in low spatial resolution images is a combination of various endmembers [5]–[9] so that it can be unmixed using the mixing theory. In weight function-based algorithms, fusion images are generated using input images through weight functions [10]–[19]. A typical example is the spatial and temporal adaptive reflectance fusion model (STARFM) [10]. Bayesian-based algorithms use Bayesian estimation theory to generate fusion images [20]–[22]. The key to Bayesian-based algorithms is to model the relationship between observed and unobserved images. Learning-based algorithms fuse multisource images through machine learning methods [23], such as dictionary-pair learning [24]–[27], artificial neural networks [28]–[30], and extreme learning machines [31]. Hybrid spatiotemporal data fusion algorithms integrate at least two of the above methods to obtain fusion images [32]–[37].

Manuscript received November 18, 2020; revised February 8, 2021 and April 20, 2021; accepted May 9, 2021. This work was supported in part by the National Key Research and Development Program of China under Grant 2019YFB2102903 and in part by the National Natural Science Foundation of China under Grant 41801306, Grant 42022060, and Grant U1711267. *(Corresponding author: Qingfeng Guan.)*

Huan Gao, Qingfeng Guan, Xue Yang, Yao Yao, Wen Zeng, and Xuantong Peng are with the School of Geography and Information Engineering, China University of Geosciences, Wuhan 430078, China, and also with the National Engineering Research Center of GIS, China University of Geosciences, Wuhan 430078, China (e-mail: ghcug14@cug.edu.cn; guanqf@cug.edu.cn; aurora@cug.edu.cn; yaoy@cug.edu.cn; zengwen@cug.edu.cn; 994684265@qq.com).

Xiaolin Zhu is with the Department of Land Surveying and Geo-Informatics, The Hong Kong Polytechnic University, Hong Kong (e-mail: xlzhu@polyu.edu.hk).

Digital Object Identifier 10.1109/TGRS.2021.3080384

The flexible spatiotemporal data fusion (FSDAF) is a hybrid spatiotemporal data fusion algorithm and uses an unmixing analysis and a thin plate spline (TPS) interpolator to generate images with high spatial and temporal resolutions [34]. FSDAF is suitable for heterogeneous landscapes and can effectively capture land cover changes. In recent years, FSDAF has been used in a number of applications, such as monitoring dynamics of impervious surface [38], wetland [39], land surface temperature [40], and vegetation [41]. Moreover, FSDAF provides a framework for addressing both gradual and abrupt land-cover changes during the spatiotemporal fusion process. Several improved variants based on FSDAF have been developed in the last few years, such as the improved FSDAF model [42]–[44], the enhanced FSDAF model [45], the enhanced FSDAF model considering subpixel class fraction change information [46], the FSDAF 2.0 [47], and the improved FSDAF to generate suspended particulate matter concentrations [48].

However, FSDAF and many other spatiotemporal data fusion algorithms have focused on the accuracy of fused images rather than the computational efficiency of algorithms. When dealing with a large quantity of data, the computing time of a fusion algorithm can be extensive, greatly limiting its applications in monitoring long-term and large-scale land surface dynamics. Despite using fewer input images than many other algorithms, FSDAF is still subject to computationally intensive procedures that use neighborhood information, such as TPS interpolation. In addition, the domain decomposition strategy of FSDAF leads to accuracy loss at the edges of subdomains because the pixels at edges do not have sufficient information for the neighborhood-scope procedures. Therefore, improving computational efficiency is an urgent task for promoting the practical value of FSDAF and other spatiotemporal fusion methods.

Parallel computing, which uses multiple processing units to collaborate on a common task [49], is a promising solution for processing massive remotely sensed data [50], [51]. In parallel computing, the computing task is decomposed into subtasks, which can be processed simultaneously by multiple processing units. The key to parallelization is whether the computing task can be divided and carried out concurrently. Raster is the primary data structure for remote sensing images. Raster data are typically organized as matrices of pixels, which can be divided into groups and processed in parallel; raster data are thus highly suitable for parallel computing. In recent years, the advancement of geospatial technologies has generated large-scale geospatial databases with high spatial resolutions, such as the global land cover mapping at a 30-m resolution [52], the SEN12MS dataset with resolutions of 10–500 m [53], and the FROM-GLC10 dataset at a 10-m resolution [54]. Meanwhile, with the introduction of advanced statistical and machine learning techniques for the processing and analysis of remote sensing images, remote sensing algorithms are becoming more complex and computationally intensive. High data intensity and computational intensity greatly increase computing capacity requirements; therefore, parallel computing has been used in many geospatial algorithms and models for better efficiency, such as cluster analysis [55], [56], spatial interpolation [57], [58], change detection [59], relative radiometric normalization [60], urban growth simulation [61], [62], remote sensing image classification [63], and watershed modeling [64].

A graphics processing unit (GPU) is a processor for rendering computer graphics [65]. With rapid performance and capability advancements, modern GPUs cannot only handle graphics processing tasks but also general-purpose computation [66], [67]. A general-purpose graphics processing unit (GPGPU) is a GPU that capable of processing general-purpose computation. Compared with central processing units (CPUs), GPGPUs have much higher memory bandwidths and more computing cores, thus exhibiting highly improved computing performance. They have been used for the processing and analysis of remotely sensed data, including hyperspectral image classification [68], hyperspectral unmixing [69], target detection [70], and compressive sensing [71]. Therefore, parallelization on GPUs is a promising solution for overcoming the computational constraints of FSDAF and improving computational performance, in addition to the feasibility and scalability of FSDAF, especially in reference to large-scale applications.

To address the aforementioned limitations of FSDAF, this article proposes an enhanced FSDAF algorithm parallelized using GPUs, named cuFSDAF; the objective is to improve the computational efficiency while maintaining the accuracy. In cuFSDAF, the TPS interpolator is replaced by an accelerated inverse distance weighted (IDW) interpolator to reduce computational complexity. The computationally intensive procedures are parallelized using the compute unified device architecture (CUDA), a parallel computing framework for GPUs. Moreover, an adaptive domain decomposition (ADD) method is proposed to adaptively adjust the size of subdomains according to the hardware properties and ensure accuracy at the edges of subdomains. Real satellite images were used to assess the performance of cuFSDAF, and the results were compared with those of the original FSDAF and the sensor-bias driven spatio-temporal fusion (BiaSTF) model based on convolutional neural networks [30], the latest deep-learning-based spatiotemporal fusion algorithm.

II. Method

A. Brief Introduction to FSDAF

As shown in Fig. 1, FSDAF requires a pair of images at t_1 and an image with a low spatial resolution (hereafter called the coarse image) at t_2 as the input data, and the output is an image with a high spatial resolution (hereafter called the fine image) at t_2. The image pair at t_1 includes one fine image and another coarse image from different sensors.

FSDAF includes four main steps: 1) predicting a fine image at t_2 using unmixing analysis; 2) predicting a fine image at t_2 by TPS interpolation; 3) distributing the residuals of two predicted images; and 4) mitigating errors using neighborhood information.

In the first step, FSDAF assumes that each pixel in a fine image (hereafter called the fine pixel) is an endmember, and a pixel in a coarse image (hereafter called the coarse pixel)

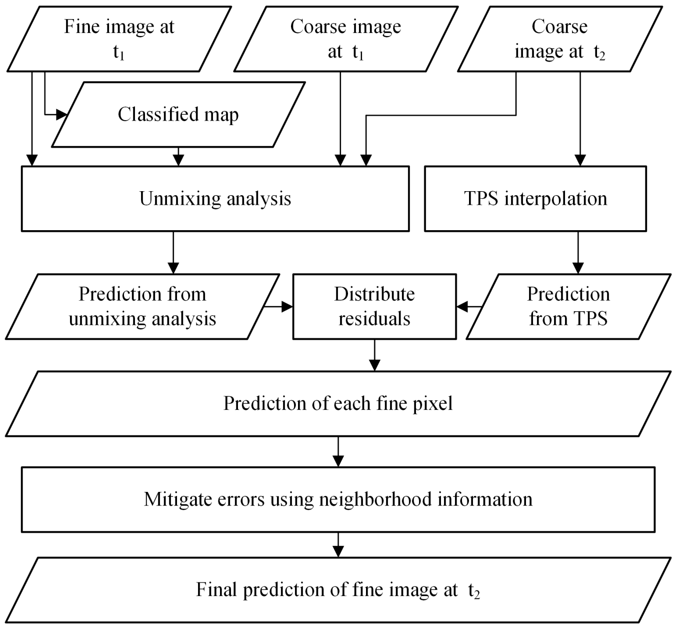

Fine image at t1
Coarse image at t1
Coarse image at t2
Classified map
Unmixing analysis
TPS interpolation
Prediction from unmixing analysis
Distribute residuals
Prediction from TPS
Prediction of each fine pixel
Mitigate errors using neighborhood information
Final prediction of fine image at t2

Fig. 1. Flowchart for the FSDAF algorithm (modified from [34]).

consists of multiple fine pixels. According to linear mixing theory, the reflectance of a coarse pixel is

$$C(x_j, y_j) = \frac{1}{N}\sum_{i}^{N} F(x_i, y_i) + \xi \quad (1)$$

where $C(x_j, y_j)$ and $F(x_i, y_i)$ are the reflectance of the coarse pixel (x_j, y_j) and fine pixel (x_i, y_i), N is the number of fine pixels inside (x_j, y_j), and ξ is the systematic difference between the two sensors. The unmixing analysis assumes that the temporal changes of fine pixels with the same class are equal and that no land-cover changes occur from t_1 to t_2. Therefore, the temporal change of the coarse pixel (x_j, y_j) from t_1 to t_2 can be represented by the mixing equation

$$C_{t_2}(x_j, y_j) - C_{t_1}(x_j, y_j) = \sum_{k}^{N_c} \Delta F(k) \times f_k(x_j, y_j) \quad (2)$$

where N_c is the number of classes in (x_j, y_j), $\Delta F(k)$ is the reflectance change of land-cover class k from t_1 to t_2, and $f_k(x_j, y_j)$ is the fraction of class k in the coarse pixel (x_j, y_j). $\Delta F(k)$ can be calculated by solving the set of mixing equations in (2), and the prediction of a fine pixel (x_i, y_i) is its reflectance at t_1 plus the temporal change of class k if it belongs to class k

$$F'_{t_2}(x_i, y_i) = F_{t_1}(x_i, y_i) + \Delta F(k) \quad (3)$$

where $F'_{t_2}(x_i, y_i)$ is the prediction of fine pixel (x_i, y_i) using unmixing analysis and $F_{t_1}(x_i, y_i)$ is the reflectance of (x_i, y_i) at t_1.

The TPS interpolator is a spatial interpolation method for point data based on spatial dependence [72]. In FSDAF, TPS is used to capture land-cover changes and local variability. The function of TPS is

$$f_{\text{tps}}(x_i, y_i) = a_0 + a_1 x_i + a_2 y_i + \frac{1}{2}\sum_{j=1}^{N} b_j r_j^2 \log r_j^2 \quad (4)$$

where $f_{\text{tps}}(x_i, y_i)$ is the prediction of a fine pixel (x_i, y_i), N is the number of known points, and r_j is the Euclidean distance between (x_i, y_i) and the jth known point (x_j, y_j). As mentioned above, one coarse pixel consists of multiple fine pixels and has one known point. In FSDAF, the known point of a coarse pixel is the central fine pixel within the coarse pixel.

These two predictions have their own disadvantages. Unlike the basic assumptions of unmixing analysis, land-cover changes and within-class variation in real-world applications cause residuals between the prediction of unmixing analysis and true reflectance. The interpolator behaves well in the homogeneous area, whereas the interpolation result is too smooth to represent spatial details in heterogeneous landscapes. In the third step, FSDAF distributes the residuals of prediction using unmixing analysis with the guidance of TPS prediction and homogeneity of landscapes. The prediction of the fine pixel (x_i, y_i) is the sum of the prediction using unmixing analysis and the distributed residual

$$F_{t_2}(x_i, y_i) = F'_{t_2}(x_i, y_i) + r(x_i, y_i) \quad (5)$$

where $F_{t_2}(x_i, y_i)$ is the prediction of the fine pixel (x_i, y_i), and $r(x_i, y_i)$ is the distributed residual.

Like STARFM [10] and ESTARFM [13], the final step of FSDAF uses neighborhood information to mitigate the uncertainty resulting from previous computing procedures and noise in input images. The final prediction of a fine pixel (x_i, y_i) is the weighted average prediction of its surrounding similar pixels

$$\bar{F}_{t_2}(x_i, y_i) = \frac{1}{\sum_k^N w_k} \times \sum_{k}^{N} w_k \times F_{t_2}(x_i, y_i) \quad (6)$$

where $\bar{F}_{t_2}(x_i, y_i)$ is the final prediction of the fine pixel (x_i, y_i) at t_2, N is the number of similar pixels, and w_k is the weight of the kth similar pixel (x_k, y_k). The similar pixels around the target pixel have similar spectral characteristics to the target pixel. The pixel weight is associated with the spatial distance between the target pixel and the similar pixel.

One of the key limitations of FSDAF that prevents its use in large-scale applications and mass production is its extensive computational intensity. As described above, the procedures of FSDAF are complicated and computationally expensive. First, the time complexity of TPS is $O(n^3)$, given that the number of known points is n [73]. To make TPS computationally feasible, it has been suggested that the number of known points should not be larger than 2000 [74]. However, when using large-sized input images, the number of known points can easily exceed 2000, leading to extremely extensive computing time for the TPS interpolation, which greatly reduces the feasibility and applicability of FSDAF. Therefore, FSDAF splits the entire spatial domain into multiple subdomains and processes them one at a time. By manually setting the maximum size, each subdomain can be small enough to process within a feasible period. Nevertheless, even with the domain decomposition strategy, FSDAF is still computationally intensive, given the computational complexity of TPS and a large number of subdomains to be processed when dealing with large images.

Second, several procedures in FSDAF are implemented using a moving window to acquire neighborhood information, leading to high demands for computing resources. For instance, the last step of FSDAF is to ensure the spatial continuity of fused images using neighborhood information. Although such a strategy can effectively reduce the uncertainties of fusion results, significant levels of computation are required. Similarly, other procedures using neighborhood information in FSDAF, such as the TPS interpolation and calculating the homogeneity of pixels, are also computationally intensive [34].

In addition to computational intensity, the domain decomposition strategy of FSDAF means that certain procedures using neighborhood information fail to acquire sufficient neighboring pixels for the target pixels at the edges of subdomains; therefore, the fusion results at the edges of subdomains may be less accurate.

B. cuFSDAF

The enhanced FSDAF algorithm parallelized using CUDA (cuFSDAF) proposed in this study is enhanced as follows: 1) the TPS interpolator is replaced by an accelerated IDW interpolator to reduce computational complexity; 2) the algorithm is parallelized based on CUDA to utilize multithreading of GPUs; and 3) an ADD method is proposed to improve the fusion accuracy at the edges of subdomains and to enable various GPUs to handle datasets of any size.

As shown in Fig. 2, the heterogeneous parallel computing framework is adopted in cuFSDAF, which includes a CPU and a GPU. The data input and output (I/O) are handled by the CPU, and the computing procedures are carried out by the CPU or GPU. Procedures with low computational intensities (e.g., unmixing analysis and residual distribution) are handled by the CPU, and the GPU manages parallelizable and computationally expensive procedures (i.e., interpolation, calculating the homogeneity of pixels, and mitigating errors using neighborhood information). Before the actual computation, the input images are decomposed adaptively into subdomains according to the device properties of GPU (e.g., memory size) and the dimensions of the input images, such that any CUDA-enabled GPU can be utilized in its maximum capacity to handle a dataset of any size.

1) Accelerated IDW Interpolator: In FSDAF, the TPS interpolator is used to capture the spatial details of land cover changes and local variability [34]. The FSDAF uses a domain decomposition strategy to make the TPS feasible. However, given the computational complexity of TPS, the total computing time is particularly lengthy for large images. In addition, the matrix inversion procedure in TPS requires extensive computation and is difficult to parallelize. Therefore, an accelerated IDW interpolator is used in cuFSDAF to replace the TPS interpolator.

The IDW interpolator assumes that each pair of points is related to each other [75], and the relevance corresponds to their distance apart. For a target point (x_0, y_0), the interpolation

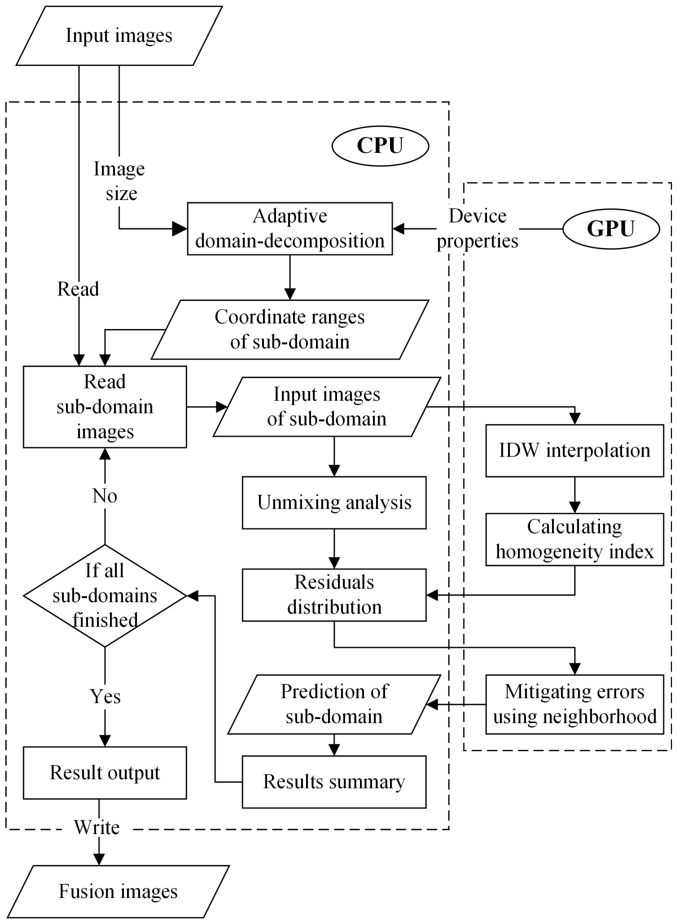


Fig. 2. Flowchart for the cuFSDAF algorithm.

result is the weighted sum of every known point

$$f_{\text{IDW}}(x_0, y_0) = \sum_{i=1}^{N} w_i f(x_i, y_i) \tag{7}$$

where N is the number of known points. Similar to FSDAF, the known point of a coarse pixel in cuFSDAF is the central fine pixel within the coarse pixel. w_i is the weight of the known point (x_i, y_i) and is often defined as

$$w_i = \frac{d_i^{-n}}{\sum_{i=1}^{N} d_i^{-n}} \tag{8}$$

where d_i is the Euclidean distance between the target point (x_0, y_0) and the known point (x_i, y_i), and n is a positive power parameter with recommended values of 1–3. The optimal value of n can be determined by the homogeneity of the landscapes. In general, higher values (e.g., >2) can be used for areas with high heterogeneity to reserve neighboring spatial details. In practice, the value of n can be determined by comparing the interpolated result with the fine image reserved for validation purposes.

For known points far from a target point, the weights are often too light. To enhance the computational efficiency, cuFSDAF only considers those known points within a certain distance r from the target point. This strategy reduces the computation, but the distances between the target point and every known point should be calculated. In cuFSDAF, this traversal operation was avoided by preliminary selection. For the target point (x_0, y_0), the known points are located in a circle with a radius of r, and the coordinate ranges of the minimum enclosing square of the circle are calculated as

$$x_i \in [x_0 - r\ ,\ x_0 + r] \tag{9}$$

and

$$y_i \in [y_0 - r\ ,\ y_0 + r]. \tag{10}$$

Next, the known points of the square are selected for distance calculation. Through such a modification, the time complexity of the IDW interpolator is O(n), where n is the number of target points.

The IDW interpolation result may not be as accurate as that of TPS. The TPS interpolation is replaced by the accelerated IDW for several reasons. First, IDW can generate high-quality interpolation results when using evenly distributed known points [75], which is the case with FSDAF/cuFSDAF. Second, compared to TPS, whose time complexity is O(n^3), the accelerated IDW, with time complexity O(n), is much more efficient and therefore suited to large-scale interpolation tasks. Third, IDW can be easily parallelized, whereas the matrix inversion in TPS makes parallelization problematic. Although Sharma *et al.* [76] parallelized matrix inversion using CUDA, this requires n^2 threads, given that n is the size of the matrix. In large-scale applications for land cover dynamics, the matrix size can be too large for a GPU to provide sufficient threads; thus, parallel matrix inversion is still infeasible. In contrast, the IDW interpolation for a particular point is independent of the interpolations for other points; thus, the computing task can be compartmentalized easily into subtasks and processed simultaneously.

2) Parallelization on GPU: Several procedures in cuFSDAF require information about the neighborhood of each target pixel. For example, the IDW interpolation requires the reflectance of its neighboring pixels when estimating the value of the target pixel. In addition, the homogeneity index (HI) is calculated to measure the homogeneity of a fine pixel and is used for the distribution of residuals. In cuFSDAF, HI equals the fraction of neighboring pixels with the same land cover type, and this fraction is calculated according to the land cover type of neighboring pixels around the target. Moreover, the weighted average of the predictions for neighboring pixels with similar spectral characteristics is used to further mitigate errors in the fused images. Although these procedures are computationally intensive and require extensive computing time when dealing with large images, they are all parallelizable, as the computation for a given pixel is independent of the computations for other pixels in these procedures. General-purpose parallel frameworks, such as the CUDA by Nvidia, enable GPUs to perform general-purpose computations using multiple threads simultaneously [77], [78]. In cuFSDAF, all three procedures mentioned above (i.e., IDW interpolation, HI calculation, and error mitigation using neighboring pixels) are parallelized using GPU through CUDA.

It is important to note that the parallel procedures in cuFSDAF are separate from each other (see Fig. 2). When a procedure is finished (e.g., IDW interpolation), the GPU will reset and all threads are re-assigned for the next procedure (e.g., HI calculation). Maintaining their independence makes them portable for the parallelization of other spatiotemporal fusion algorithms if they use similar operations as FSDAF, such as those variants of FSDAF.

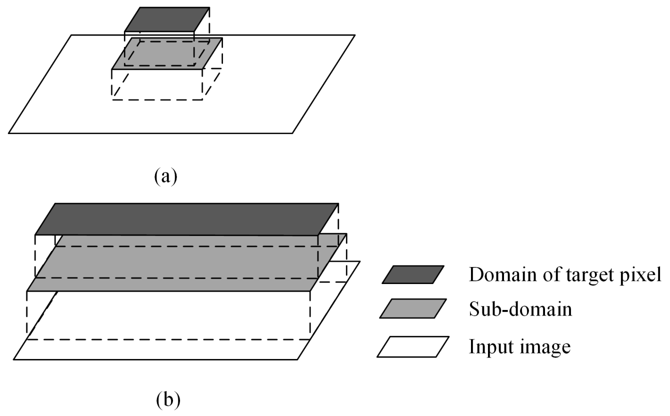


Fig. 3. Domain decomposition method in FSDAF and cuFSDAF. (a) Subdomain in FSDAF. (b) Subdomain in cuFSDAF.

3) Adaptive Domain Decomposition: Domain decomposition is still necessary for cuFSDAF, especially when dealing with large images and the GPU memory is not sufficient to accommodate all of the data at once. An ADD method is proposed for cuFSDAF.

Compared with the original domain decomposition method in FSDAF, ADD has two enhancements. First, the maximum size of the subdomain is determined automatically according to the hardware specifications of the GPU (e.g., the available video memory). The domain decomposition method in FSDAF decomposes images into squares with a user-specified size [Fig. 3(a)]. The ADD in cuFSDAF uses a row-wise decomposition strategy and divides the domain into rectangular subdomains with widths equal to the widths of the input images. The maximum height of a subdomain is determined adaptively, such that a subdomain contains as many pixels as the GPU can handle at one time [Fig. 3(b)]. In other words, cuFSDAF can automatically adapt to GPUs with different memory capacities and maximize the memory utilization and computing capacity of the GPU. In general, the subdomain size determined by the ADD of cuFSDAF is much larger than the size used in FSDAF, resulting in fewer subdomains and, therefore, quicker data transfer between the CPU and GPU.

The second enhancement is the adaptive determination of the neighborhood size. For target pixels at the edges of subdomains, FSDAF extends extra "halo" pixels [49] for the TPS interpolator to preserve edge details [79]; thus, a subdomain is larger than the block of target pixels [see Fig. 3(a)]. However, this strategy does not include sufficient neighboring halo pixels of an edge pixel for other neighborhood-scope procedures besides interpolation (e.g., HI calculation and error mitigation using neighborhood pixels), which may undermine the fusion accuracy for pixels on the edges of subdomains.

A subdomain generated by the ADD of cuFSDAF not only holds the target pixels for processing but also holds valuable neighboring pixels (i.e., halo pixels) for all

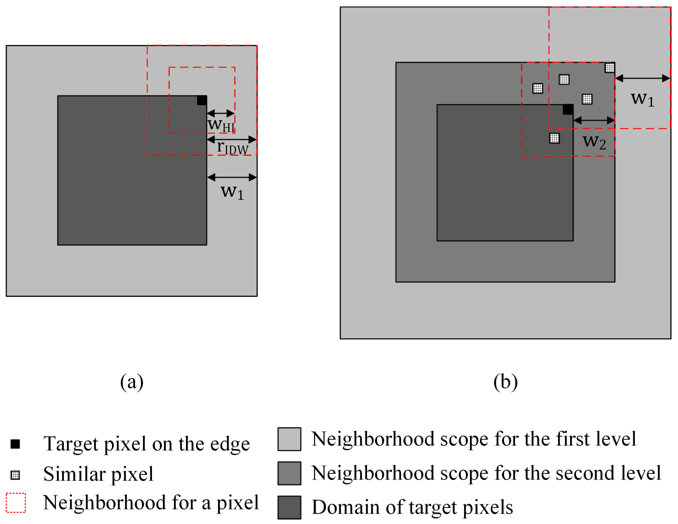


Fig. 4. Two-level neighborhood at subdomain edges. (a) Neighborhood for the first level. (b) Neighborhood for the second level.

neighborhood-scope procedures [see Fig. 3(b)]. In cuFSDAF, the width of the neighborhood is calculated adaptively, according to the demands of neighborhood-scope calculations and unmixing analysis. Two levels of neighborhood-scope calculations exist in the cuFSDAF. The first level includes the IDW interpolator and HI calculation [see Fig. 4(a)]. To ensure that all target pixels can retrieve necessary neighborhood information, the width of the neighborhood is the maximum between the searching radius of IDW and the neighborhood width for HI calculation

$$w_1 = \max\{r_{\text{IDW}}, w_{\text{HI}}\}. \tag{11}$$

A neighborhood with width w_1 ensures accurate prediction before error mitigation using neighborhood information.

The second level of neighborhood-scope calculation is to mitigate errors. As shown in Fig. 4(b), the width of the window for searching similar pixels is w_2, and the potential farthest similar pixel may be located at the edge of the neighborhood. To ensure the accuracy of the first-level calculations for these pixels, the total neighborhood width needs to expand w_1. Therefore, the width of the neighborhood for neighborhood-scope calculations should be the sum of the widths of the two levels

$$w_n = w_1 + w_2. \tag{12}$$

Moreover, the total width of the neighborhood in cuFSDAF may be larger than the neighborhood width for neighborhood-scope calculations, because w_n may not be enough to mitigate the block effect effectively if the subdomain size is too large. The unmixing analysis in FSDAF is independent across subdomains, thus the pixels on different sides of a subdomain edge will have different temporal changes, despite belonging to the same class of endmembers. Extending "halo" pixels regarding the subdomain size can reduce such block effect resulted from the unmixing analysis. The neighborhood width for unmixing analysis is

$$w_u = aW \tag{13}$$

where W is the height of subdomains, and a is the ratio of W ranging from 0 to 1. Therefore, the width of the neighborhood in cuFSDAF equals the maximum between the neighborhood width for neighborhood-scope calculations and the width for unmixing analysis

$$w = \max\{w_n, w_u\}. \tag{14}$$

III. Experiments

The cuFSDAF was implemented using the C++ programming language and CUDA, and the source code is publicly available at https://github.com/HPSCIL/cuFSDAF. To provide the baselines for accuracy and efficiency assessments, a serial FSDAF was implemented using C++, which can generate the same results as the IDL-implemented FSDAF (https://xiaolinzhu.weebly.com/open-source-code.html), but 1.6–2.0 times faster on a workstation computer equipped with an Intel Xeon W-2133 CPU @ 3.6 GHz and 16 GB of main memory. To evaluate the performance of cuFSDAF in large-scale and long-term spatiotemporal data fusion tasks, one of the newest deep-learning-based spatiotemporal fusion algorithms, the BiaSTF model based on convolutional neural networks [30], was also used in the experiments for comparison.

A. Testing Dataset and Environment

To assess the prediction accuracy and computational performance of cuFSDAF, we selected three sets of satellite images (Table I and Fig. 5) from the benchmark datasets for spatiotemporal fusion provided by Li *et al.* [80], including the AHB, Tianjin, and Daxing datasets. Each dataset includes eight pairs of images, and each pair consists of a MODIS and a Landsat image as the coarse and fine images. The time gap between the first image pair and the last pair in each dataset is around 1.5–2 years. The necessary atmospheric correction, geometric transformation, resampling, and band rearrangement were applied to these datasets. Considering the strip noises in the short-wave infrared bands in the MODIS images [80], we used four bands (i.e., blue, green, red, and near-infrared band of Landsat 8 OLI and their corresponding bands of MODIS) from these images.

Except for the unique parameters of cuFSDAF (i.e., searching radius and power for IDW), the same parameter settings were used for both cuFSDAF and FSDAF. For BiaSTF, we used the parameters recommended by the authors of BiaSTF. In our experiments, we used the image pairs 1–5 and 8 for CNN training and predicted fine images on the dates of image pairs 6 and 7. In cuFSDAF and FSDAF, a pair of images on the base date, a coarse image on the prediction date, and a classified image by the ISODATA classifier [81] (based on the fine image on the base date) were used as the input to predict a fine-resolution image on the prediction date. BiaSTF requires a former image pair and a later image pair for a fusion image [30]. Therefore, we picked one pair closest to the fusion date as the input image pair for FSDAF/cuFSDAF, and picked two pairs closest to the fusion date as input images for BiaSTF. For instance, when generating the fusion image on July 7,

TABLE I
TEST DATASETS FOR COMPARATIVE EXPERIMENTS

	AHB	Tianjin	Daxing
Study area	Ar Horqin Banner of Inner Mongolia province, China	Tianjin municipality, China	Daxing district of Beijing city, China
Area characteristics	Heterogeneous area with phenological changes	Urban area with phenological changes	Area with land-cover changes
Dates of image pairs	2014/4/15 2014/9/6 2015/3/17 2015/5/4 2015/6/21 2015/7/7 2015/9/25 2016/4/20	2017/7/10 2017/9/12 2017/11/15 2018/2/3 2018/4/8 2018/10/1 2018/12/4 2019/1/21	2017/7/10 2017/9/12 2017/10/30 2017/11/15 2018/2/3 2018/4/8 2018/10/1 2018/12/4
Image size	2480×2480×4	1920×1920×4	1640×1640×4

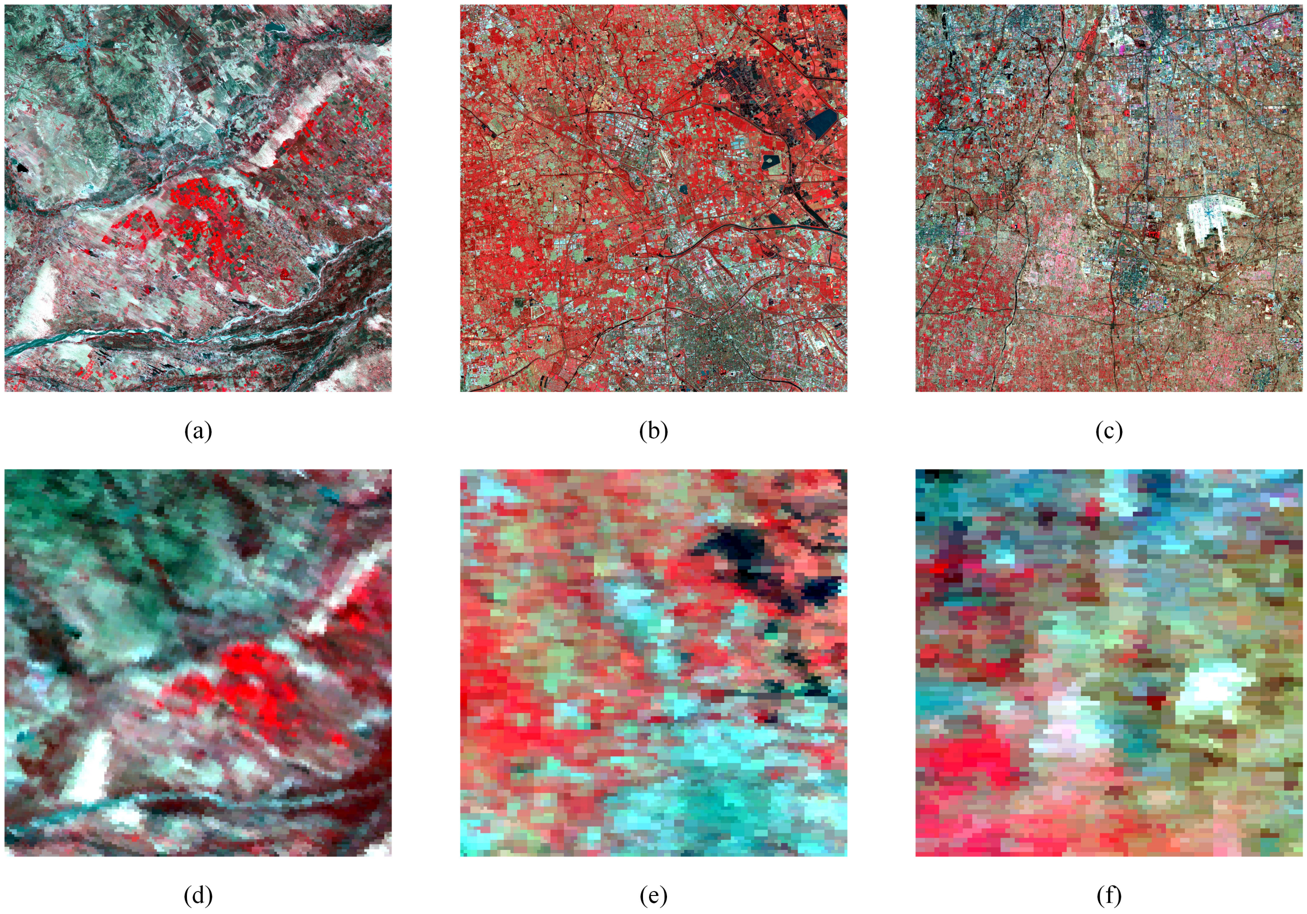


Fig. 5. Image pairs of three datasets. (a) Landsat image on June 21, 2015 in the AHB dataset. (b) Landsat image on October 1, 2018 in the Tianjin dataset. (c) Landsat image on April 8, 2018 in the Daxing dataset. (d) MODIS image on June 21, 2015 in the AHB dataset. (e) MODIS image on October 1, 2018 in the Tianjin dataset. (f) MODIS image on April 8, 2018 in the Daxing dataset.

2015, using the AHB dataset, the image pairs on June 21, 2015, and September 25, 2015, were input to BiaSTF, and the image pair on June 21, 2015, were input to both FSDAF and cuFSDAF.

The resultant image was compared visually and quantitatively with the corresponding true images. The accuracy indices used in the experiments include the root mean square error (RMSE), the correlation coefficient (CC), the structure similarity (SSIM) [82], the spectral angle mapper (SAM) [83], and the erreur relative globale adimensionnelle de synthese (ERGAS) [84]. Besides, the computing time of each experiment was recorded as an indicator of computational performance. Additional experiments were conducted using the AHB dataset to compare the fusion result by TPS with that by IDW, to assess the variation at the edges of subdomains with and without the proposed ADD method, and to compare cuFSDAF and FSDAF when using input images with different time intervals.

The same testing environments (hardware and software) were used for all experiments. The experiments were conducted on a workstation computer equipped with an Intel Xeon W-2133 CPU @ 3.6 GHz, and a Nvidia GeForce GTX 1080ti GPU with 3584 CUDA cores and 11 GB of video memory. Other hardware and software information is shown in Table II.

B. Experimental Results

Fig. 6 shows the fusion results using the AHB dataset including the actual Landsat image [Fig. 6(a) and (e)],

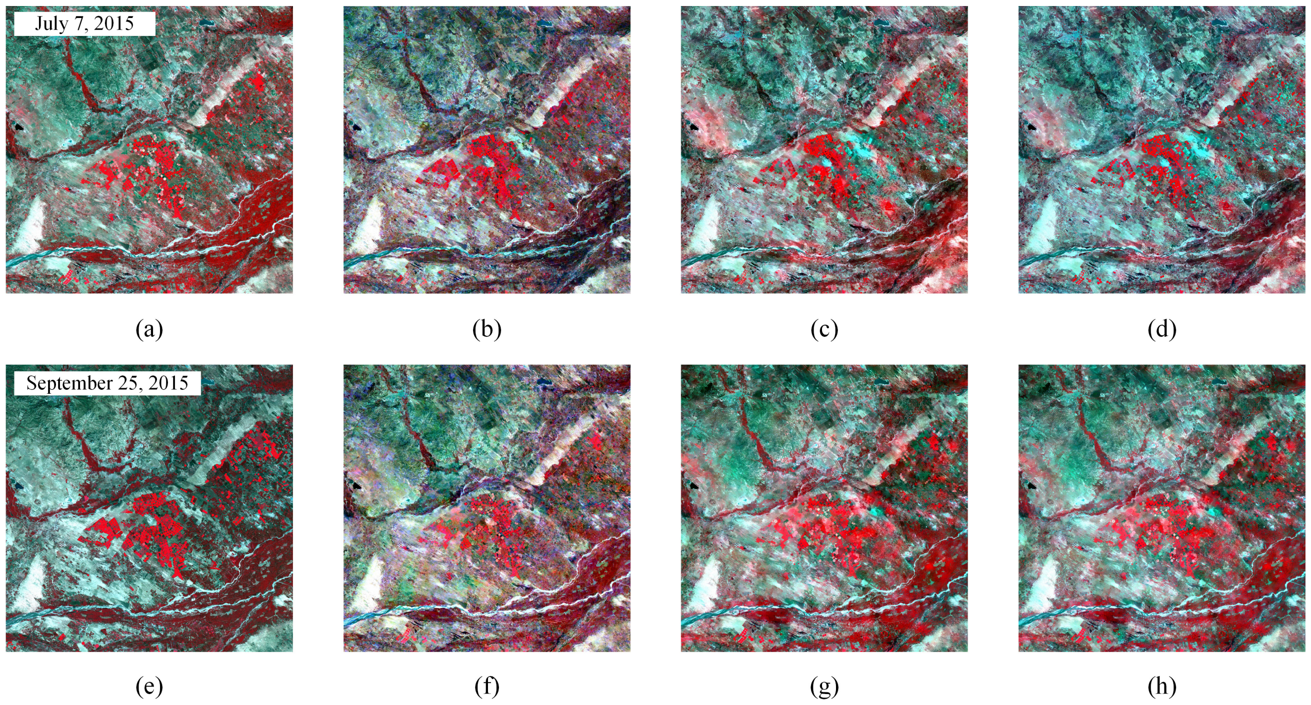


Fig. 6. Actual and fusion images for the AHB dataset: Landsat image on (a) July 7, 2015 and (e) September 25, 2015, fusion image by BiaSTF on (b) July 7, 2015 and (f) September 25, 2015, fusion image by FSDAF on (c) July 7, 2015 and (g) September 25, 2015, fusion image by cuFSDAF on (d) July 7, 2015 and (h) September 25, 2015.

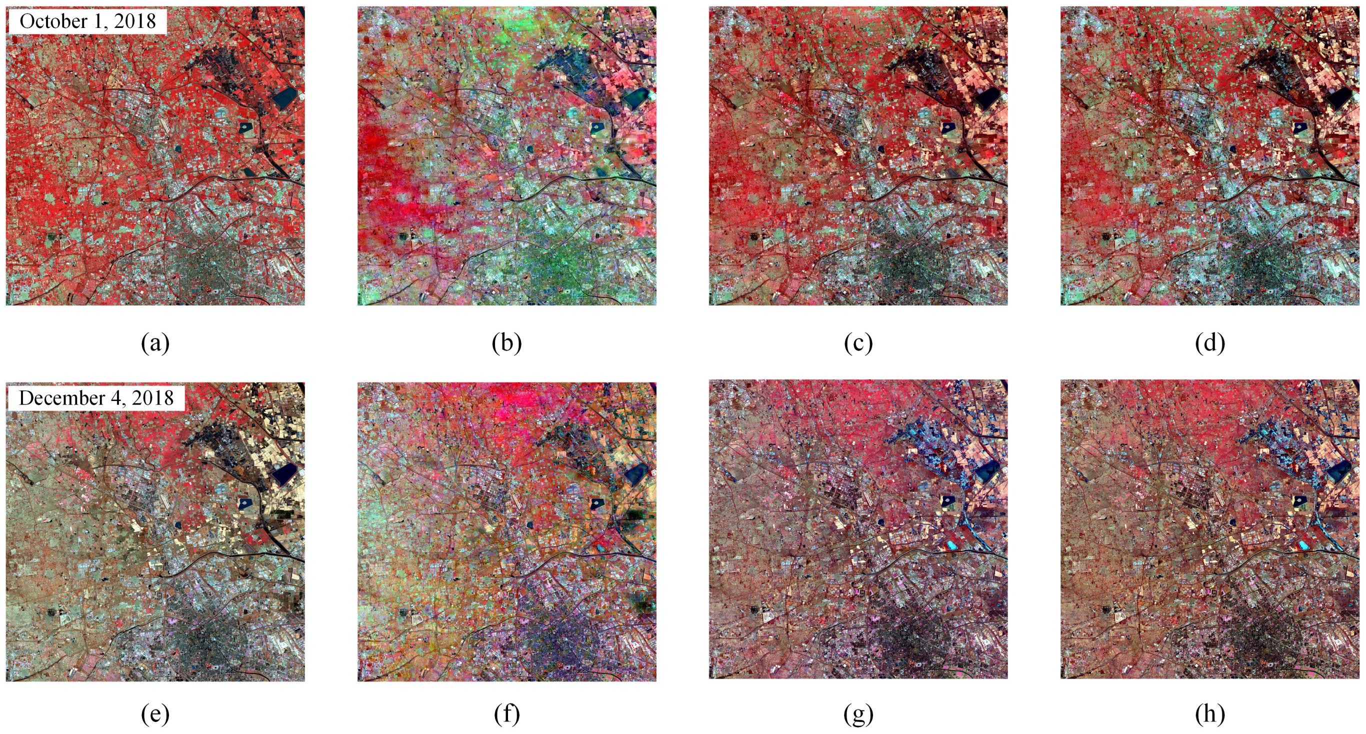


Fig. 7. Actual and fusion images for the Tianjin dataset: Landsat image on (a) October 1, 2018 and (e) December 4, 2018, fusion image by BiaSTF on (b) October 1, 2018 and (f) December 4, 2018, fusion image by FSDAF on (c) October 1, 2018 and (g) December 4, 2018, fusion image by cuFSDAF on (d) October 1, 2018 and (h) December 4, 2018.

fusion images by BiaSTF [Fig. 6(b) and (f)], FSDAF [Fig. 6(c) and (g)], and cuFSDAF [Fig. 6(d) and (h)]. The results of the Tianjin and Daxing datasets are shown in Figs. 7 and 8; quantitative indices for all three datasets are shown in Table III. Both visual comparison and quantitative analysis indicated that the accuracy of cuFSDAF is very similar to those of FSDAF and BiaSTF.

The quantitative indices of FSDAF and cuFSDAF in Table III vary slightly. Two reasons may have caused the variations. The first reason is related to the unmixing analysis,

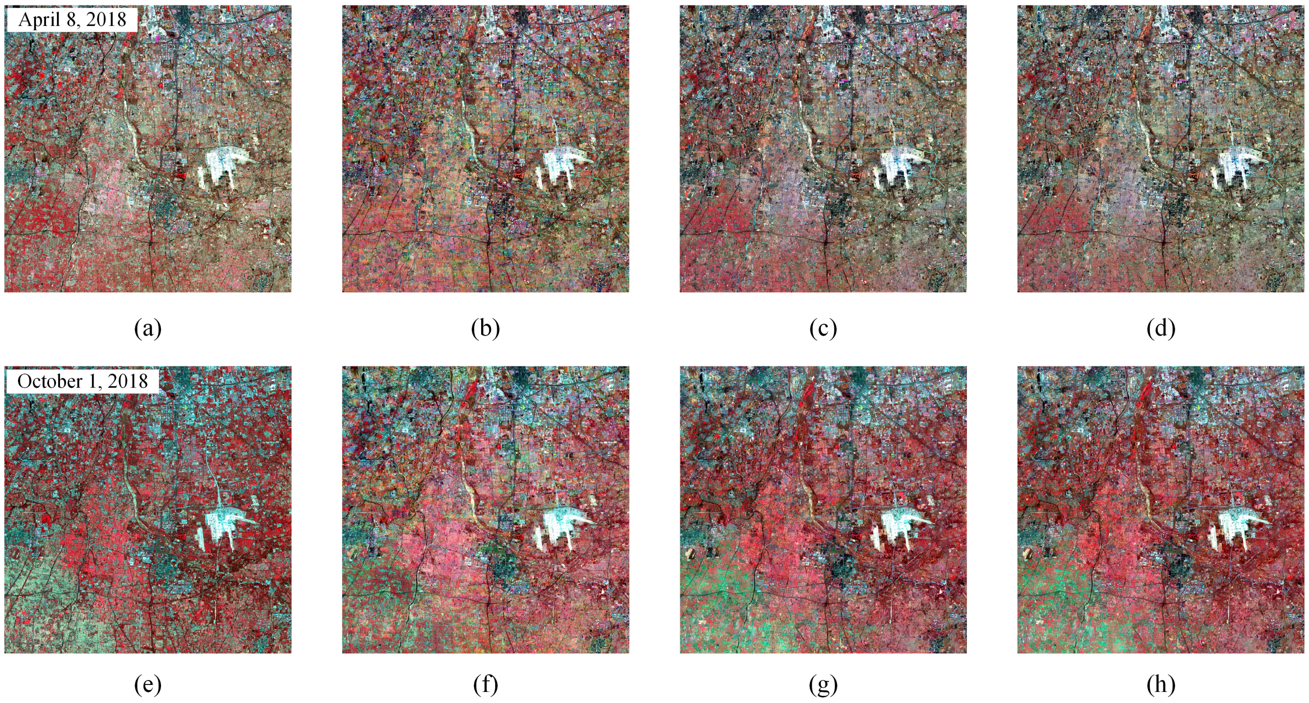


Fig. 8. Actual and fusion images for the Daxing dataset: Landsat image on (a) April 8, 2018 and (e) October 1, 2018, fusion image by BiaSTF on (b) April 8, 2018 and (f) October 1, 2018, fusion image by FSDAF on (c) April 8, 2018 and (g) October 1, 2018, fusion image by cuFSDAF on (d) April 8, 2018 and (h) October 1, 2018.

TABLE II
HARDWARE AND SOFTWARE ENVIRONMENTS FOR EXPERIMENTS

CPU	Intel Xeon W-2133 @3.6GHz
GPU	Nvidia GeForce GTX 1080ti
Main memory	12 GB
Video memory	11 GB
Operating system	Windows 10 x64
CUDA version	9.1
TensorFlow-gpu version	1.9.0

in which both FSDAF and cuFSDAF estimate the temporal changes using linear regression. Compared with the subdomains in FSDAF, the subdomains in cuFSDAF are larger. With the expansion of the subdomain size, the coarse pixels chosen for linear regression in cuFSDAF changed, which resulted in the accuracy differences. More specifically, the temporal changes of fine pixels of the same class may vary, but the unmixing analysis assumes their temporal changes are equal given that they belong to the same subdomain. Therefore, it is harder to capture the local intra-class variations when using larger-sized subdomains. The second reason is related to the ADD method of cuFSDAF, which reduces the block effects and increases the accuracy at the edge of subdomains. As shown in Table IV, replacing TPS with IDW did not result in obvious accuracy loss, which proved our analysis. Compared with the Daxing dataset, the subdomains for the AHB and Tianjin dataset had larger sizes, so cuFSDAF performed slightly worse than FSDAF due to more accuracy loss resulted from intra-class variations. On the contrary, cuFSDAF performed better for the Daxing dataset.

In cuFSDAF, the ADD method often generates larger-sized subdomains than those by the original FSDAF that may lead to some accuracy loss, but it would bring more benefits. First, the accuracy loss resulted from intra-class variations is slight and acceptable. Second, larger subdomain sizes result in quicker data transfer between the CPU and GPU, which improves computational efficiency. Third, larger size helps reduce the block effects. The larger the subdomain size, the fewer subdomains as well as slighter block effects.

The accuracies of FSDAF and cuFSDAF were as good as the accuracy of BiaSTF in our experiments. Compared with the datasets used by Song *et al.* [29], the time series of datasets we used are sparser. For instance, the AHB dataset consists of eight image pairs, and the time span of them is about two years. The LGC dataset used by Song *et al.* [29] consists of 14 image pairs, and the time span is about one year. Datasets with denser time series help BiaSTF better capture reflectance changes, which may result in better performance than FSDAF and FSDAF-like methods. Nevertheless, in many cloudy regions, the time series of satellite images may not be dense enough to show the superiority of BiaSTF. Besides, the experiments by Bernabé [70] showed that BiaSTF achieved higher accuracy than FSDAF when predicting one fusion image using two image pairs as the training data. For producing long-term time series by spatiotemporal fusion technology, retraining BiaSTF for predicting each fusion image is unlikely feasible because of the large time consumption for model training.

C. Additional Experiments

In the first additional experiment on the AHB dataset, we fused the image on September 25, 2015, using the original FSDAF with TPS (FSDAF-TPS) and the modified FSDAF with IDW (FSDAF-IDW) to evaluate the change induced

TABLE III

ACCURACY ASSESSMENTS OF THE PREDICTED IMAGES BY BIASTF, FSDAF, AND CUFSDAF. UNITS ARE REFLECTANCE. (RMSE = ROOT MEAN SQUARE ERROR, CC = CORRELATION COEFFICIENT, SSIM = STRUCTURE SIMILARITY, SAM = SPECTRAL ANGLE MAPPER, ERGAS = ERREUR RELATIVE GLOBALE ADIMENSIONNELLE DE SYNTHESE)

Dataset	Date	Model	RMSE	CC	SSIM	SAM	ERGAS
AHB	2015/7/7	BiaSTF	0.042	0.700	0.757	0.201	1.501
		FSDAF	**0.028**	**0.752**	**0.813**	**0.158**	**0.852**
		cuFSDAF	0.031	0.722	0.783	0.174	0.933
	2015/9/25	BiaSTF	**0.056**	0.668	**0.737**	0.227	**2.605**
		FSDAF	0.063	**0.702**	0.701	**0.199**	3.126
		cuFSDAF	0.064	0.699	0.698	0.200	3.128
Tianjin	2018/10/1	BiaSTF	0.043	0.775	0.769	**0.313**	2.980
		FSDAF	**0.037**	**0.786**	**0.781**	0.354	**2.270**
		cuFSDAF	0.039	0.744	0.755	0.373	2.364
	2018/12/4	BiaSTF	**0.028**	**0.824**	**0.862**	**0.226**	**1.329**
		FSDAF	0.034	0.775	0.805	0.282	1.840
		cuFSDAF	0.035	0.768	0.796	0.296	1.909
Daxing	2018/4/8	BiaSTF	0.033	**0.809**	**0.840**	**0.178**	1.125
		FSDAF	**0.027**	0.793	0.834	0.195	1.048
		cuFSDAF	**0.027**	0.803	0.835	0.196	**1.046**
	2018/10/1	BiaSTF	0.048	0.745	0.725	0.321	3.085
		FSDAF	**0.034**	0.749	0.790	0.312	1.919
		cuFSDAF	**0.034**	**0.755**	**0.795**	**0.306**	**1.869**

TABLE IV

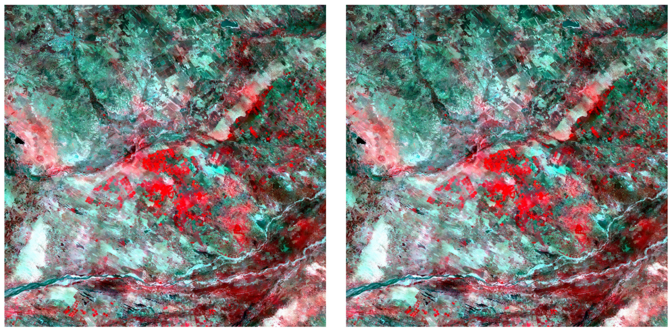
ACCURACY ASSESSMENT OF THE FUSION IMAGES BY FSDAF-TPS AND FSDAF-IDW. UNITS ARE REFLECTANCE. (RMSE = ROOT MEAN SQUARE ERROR, CC = CORRELATION COEFFICIENT, SSIM = STRUCTURE SIMILARITY, SAM = SPECTRAL ANGLE MAPPER, ERGAS = ERREUR RELATIVE GLOBALE ADIMENSIONNELLE DE SYNTHESE)

FSDAF-TPS					
	RMSE	CC	SSIM	SAM	ERGAS
Band 1	0.047	0.737	0.788	0.138	3.126
Band 2	0.050	0.777	0.758	0.186	
Band 3	0.062	0.771	0.707	0.261	
Band 4	0.095	0.524	0.549	0.209	
FSDAF-IDW					
	RMSE	CC	SSIM	SAM	ERGAS
Band 1	0.047	0.737	0.788	0.138	3.126
Band 2	0.050	0.777	0.758	0.186	
Band 3	0.062	0.771	0.707	0.261	
Band 4	0.095	0.524	0.549	0.209	

by replacing TPS with IDW. Both quantitative indices (see Table IV) and visual comparison (Fig. 9) indicate that the replacement of TPS by IDW had no obvious impact on the accuracy of the fusion results.

The second additional experiment was to evaluate the effectiveness of the proposed ADD strategy. To exclude the differences caused by the TPS and IDW interpolators, we compared the results of FSDAF with IDW using the fixed domain decomposition (FSDAF-IDW-FDD) with those obtained using the ADD strategy (FSDAF-IDW-ADD). Given that the ADD method increases the size of subdomains significantly, the edges of subdomains in FDD are no longer the edges of subdomains in ADD. To assess the performance of ADD at the edges of subdomains, the sizes of the subdomains of FSDAF-IDW-ADD were set manually, but the neighborhood width was determined using the adaptive method of ADD.

The quantitative comparisons between FSDAF-IDW-FDD and FSDAF-IDW-ADD at the edges of the subdomains are presented in Table V, which show that the accuracy was improved slightly by ADD. For better observation of the visual details, we enlarged an edge of the subdomains in



(a) (b)

Fig. 9. Fusion images using FSDAF with different interpolators. (a) Fusion image by FSDAF-TPS. (b) Fusion image by FSDAF-IDW.

TABLE V

ACCURACY ASSESSMENT OF FUSION RESULTS BY FSDAF-IDW-FDD AND FSDAF-IDW-ADD AT THE EDGES OF SUBDOMAINS. UNITS ARE REFLECTANCE (RMSE = ROOT MEAN SQUARE ERROR, CC = CORRELATION COEFFICIENT, SSIM = STRUCTURE SIMILARITY, SAM = SPECTRAL ANGLE MAPPER, ERGAS = ERREUR RELATIVE GLOBALE ADIMENSIONNELLE DE SYNTHESE)

FSDAF-IDW-FDD					
	RMSE	CC	SSIM	SAM	ERGAS
Band 1	**0.051**	**0.740**	0.774	**0.143**	3.591
Band 2	**0.054**	0.727	0.695	0.210	
Band 3	**0.072**	0.723	0.662	0.315	
Band 4	0.088	**0.656**	**0.749**	**0.188**	
FSDAF-IDW-ADD					
	RMSE	CC	SSIM	SAM	ERGAS
Band 1	**0.051**	0.739	**0.777**	**0.143**	**3.577**
Band 2	**0.054**	**0.735**	**0.705**	**0.207**	
Band 3	**0.072**	**0.734**	**0.685**	**0.309**	
Band 4	**0.086**	0.651	0.740	0.190	

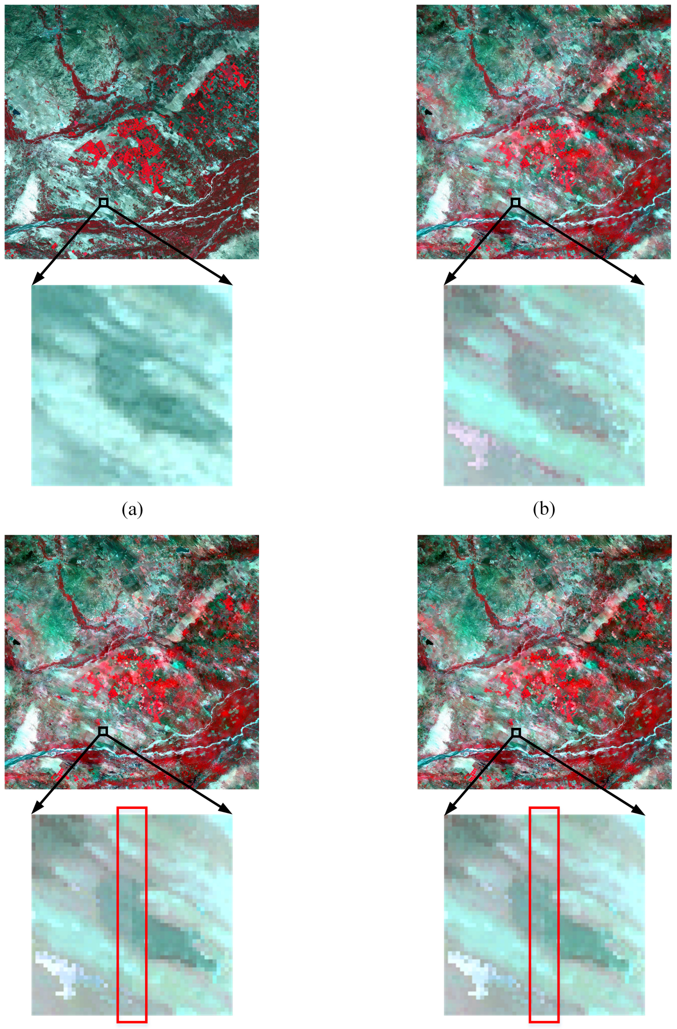


Fig. 10. Enlarged actual and fusion images from a sub-domain edge. (a) Landsat image on September 25, 2015. (b) Fusion image by cuFSDAF. (c) Fusion image by FSDAF-IDW-FDD. (d) Fusion image by FSDAF-IDW-ADD.

the fusion images by FSDAF-IDW-FDD and FSDAF-IDW-ADD. We also enlarged the same area in the Landsat image on September 25, 2015, and the fusion image by cuFSDAF for comparison (Fig. 10). Compared with FSDAF-IDW-FDD [Fig. 10(c)], the enlarged FSDAF-IDW-ADD image [Fig. 10(d)] shows that the block effects at the edges of subdomains were diminished considerably by the ADD method. As mentioned above, the pixels on different sides of a subdomain edge will have different temporal changes, despite belonging to the same class of endmembers. Therefore, the block effects at the edges of the subdomains cannot be completely resolved if domain decomposition is applied. In cuFSDAF, domain decomposition can be avoided automatically when the memory of the GPU is large enough to accommodate the entire spatial domain, in which case the block effects can be removed completely[see Fig. 10(b)].

The third additional experiment was to compare cuFSDAF and FSDAF when using input images with different time intervals. We predicted the fusion image on September 25, 2015, using different image pairs (i.e., July 7, 2015; June 21, 2015; May 4, 2015; and March 17, 2015) in the AHB dataset as the base pair of FSDAF/cuFSDAF. The time intervals of input images varied from 2 to 6 months. The quantitative indices in Table VI show that the cuFSDAF achieved similar accuracies as FSDAF.

TABLE VI

ACCURACY ASSESSMENTS OF THE PREDICTED IMAGES BY FSDAF AND CUFSDAF ON SEPTEMBER 25, 2015 USING DIFFERENT BASE PAIRS. UNITS ARE REFLECTANCE. (RMSE = ROOT MEAN SQUARE ERROR, CC = CORRELATION COEFFICIENT, SSIM = STRUCTURE SIMILARITY, SAM = SPECTRAL ANGLE MAPPER, ERGAS = ERREUR RELATIVE GLOBALE ADIMENSIONNELLE DE SYNTHESE)

			2015/7/7		
	RMSE	CC	SSIM	SAM	ERGAS
FSDAF	0.063	0.702	0.701	0.199	3.126
cuFSDAF	0.064	0.699	0.698	0.200	3.128
			2015/6/21		
	RMSE	CC	SSIM	SAM	ERGAS
FSDAF	0.066	0.612	0.633	0.236	3.175
cuFSDAF	0.067	0.602	0.621	0.241	3.194
			2015/5/4		
	RMSE	CC	SSIM	SAM	ERGAS
FSDAF	0.053	0.583	0.596	0.391	2.608
cuFSDAF	0.054	0.562	0.583	0.392	2.631
			2015/3/17		
	RMSE	CC	SSIM	SAM	ERGAS
FSDAF	0.060	0.460	0.412	0.550	3.347
cuFSDAF	0.061	0.456	0.404	0.548	3.375

D. Computational Performance

Crucially, the experiments showed that compared with FSDAF, cuFSDAF significantly reduced the computing time and improved the computational efficiency Fig. 11, while maintaining the fusion accuracy (see Table III). We fused six fusion images using the AHB, Tianjin, and Daxing datasets (two images were fused in each dataset). Without data I/O, cuFSDAF achieved speed-ups of 140.3–182.2 over the IDL-implement FSDAF, and achieved speed-ups of 84.9–93.6 over the C++-implemented FSDAF [Fig. 11(a)]. The data transmission between CPU and GPU often consumes extra time, which may affect the computational performance of cuFSDAF. For instance, the data transmission consumed 0.1, 0.1, and 0.3 s for the interpolating, calculating HI, and error mitigation when predicting the fusion image on June 21, 2015, for the AHB dataset. Considering that FSDAF has no such data transmission, the time of data transmission between CPU and GPU was included in the computing time of cuFSDAF (Fig. 11).

We also recorded the computing time for each parallel procedure in cuFSDAF and the C++-implemented FSDAF (hereafter called FSDAF). First, the CUDA-enabled parallel IDW in cuFSDAF was 1807.4, 1685.1, 1426.4, 1392.4, 1064.6, and 1105.1 times faster than the TPS in FSDAF, for the six fusion tasks, respectively [Fig. 11(b)], which induced a significant improvement in computational efficiency. Second, for calculating HI, cuFSDAF was 34.7, 35.0, 23.7, 22.2, 16.7, and 17.5 times faster than FSDAF [Fig. 11(c)]. Third, for error mitigation using neighborhood, cuFSDAF

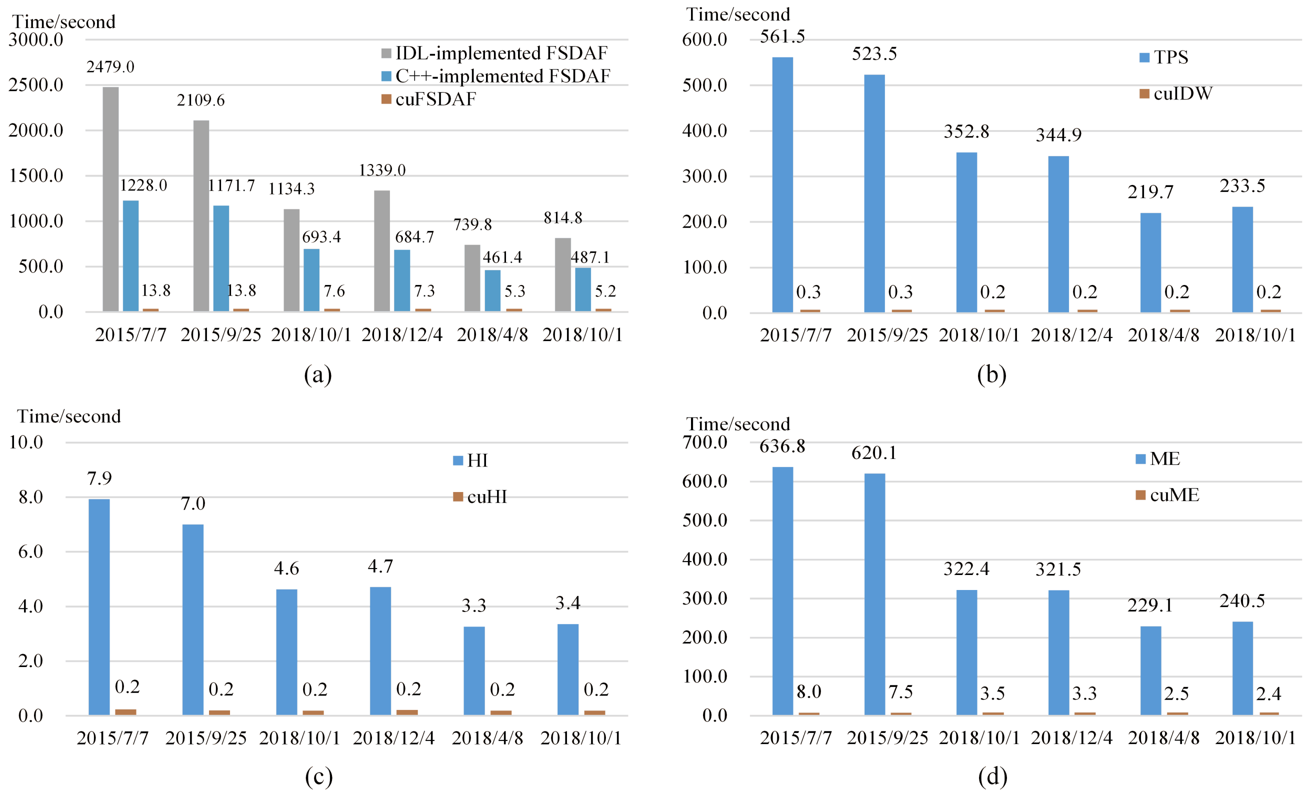


Fig. 11. Computing time for FSDAF and cuFSDAF without data I/O. (a) Total time for IDL-implemented FSDAF, C++-implemented FSDAF and cuFSDAF. (b) Time for TPS in C++-implemented FSDAF and CUDA-enabled IDW (cuIDW) in cuFSDAF. (c) Time for calculating homogeneity index in C++-implemented FSDAF (HI) and cuFSDAF (cuHI). (d) Time for error mitigation using neighborhood in C++-implemented FSDAF (ME) and in cuFSDAF (cuME).

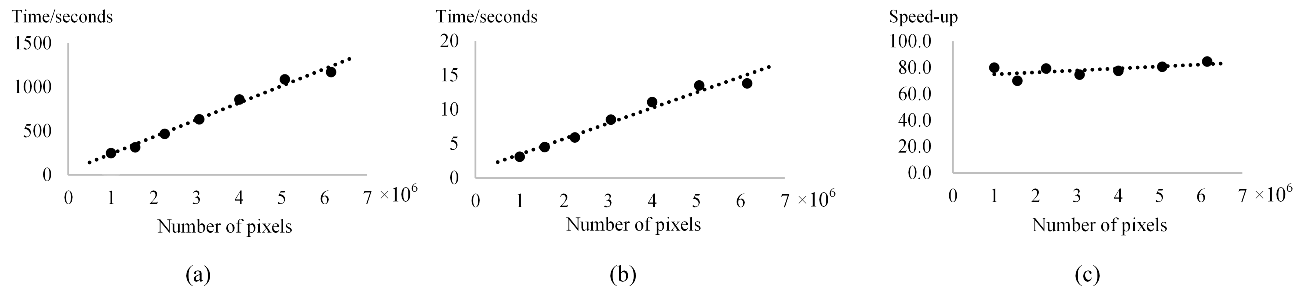


Fig. 12. Computing time and speed-ups of fusing the fusion image on September 25, 2015 using clipped AHB images with different image size. (a) Trend of computing time and number of pixels for FSDAF. (b) Trend of computing time and number of pixels for cuFSDAF. (c) Speed-ups of cuFSDAF over FSDAF.

was 79.3, 82.7, 92.1, 97.6, 91.2, and 102.0 times faster than FSDAF [Fig. 11(d)].

Notably, the speed-ups for interpolation and HI calculation were demonstrably different for the three datasets. The computing time of these procedures in cuFSDAF was too short to show differences among these three datasets, unlike FSDAF. Although the extra time for data transmission between CPU and GPU was also very short, it impacted the stability of speed-ups in these procedures more significantly than error mitigation. Besides, three procedures yielded different speedups using the same dataset. For instance, the speed-ups for interpolation, calculating HI, and error mitigation are 1807.4, 34.7, and 79.3 times when predicting the fusion image on July 7, 2015, for the AHB dataset. First, we replaced the TPS interpolator with the accelerated IDW interpolator, which greatly reduced the time complexity of the interpolation. Therefore, the modification of the interpolator achieved more significant speed-ups than other parallel procedures. Second, the computing time of interpolating and calculating HI in cuFSDAF was so short that the time of data transmission affected the speed-ups for these procedures.

We also conducted additional experiments to investigate the trend of computing time in terms of the image size by clipping the images from the AHB dataset into different sizes. As shown in Fig. 12, the computing time of both FSDAF and cuFSDAF matched the linear trend with the increasing number of pixels. The speed-up of cuFSDAF over FSDAF remained stable when the images were large enough (i.e., when the image size was equal to or greater than

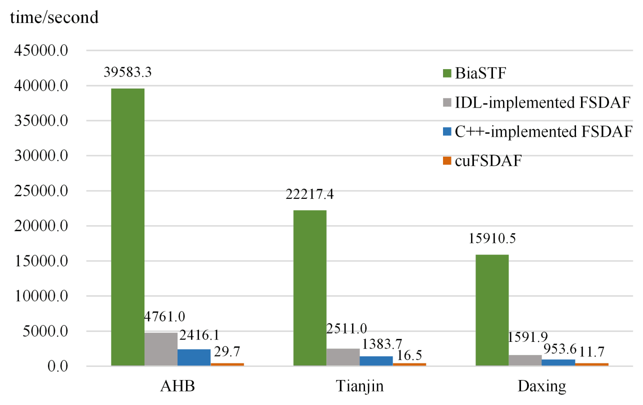


Fig. 13. Running time for BiaSTF, FSDAF, and cuFSDAF.

TABLE VII
HARDWARE AND SOFTWARE ENVIRONMENTS FOR FLEXIBILITY EXPERIMENTS

	Tianhe-2 supercomputer	Laptop computer
CPU	Intel Xeon E5-2660 @2.6GHz	Intel Core I5-7400 @3.00GHz
GPU	Nvidia Tesla K80	Nvidia Geforce GTX 1050
Main memory	256 GB	8 GB
Video memory	11,441 MB	4 GB
Operating System	Linux x86_64	Windows 10 x64
CUDA version	CUDA 8.0	CUDA 10.2

TABLE VIII
COMPUTING TIME AND SPEED-UP FOR CUFSDAF WITHOUT DATA I/O ON THE TIANHE-2 SUPERCOMPUTER AND THE LAPTOP COMPUTER

Tianhe-2 supercomputer				
Dataset	Date	FSDAF (seconds)	cuFSDAF (seconds)	Speed-ups
AHB	2015/7/7	4260.2	62.7	67.9
	2015/9/25	4240.1	60.7	69.9
Tianjin	2018/10/1	2457.4	35.1	70.0
	2018/12/4	2468.9	35.0	70.5
Daxing	2018/4/8	1640.8	26.6	61.7
	2018/10/1	1643.6	25.8	63.7
Laptop computer				
Dataset	Date	FSDAF (seconds)	cuFSDAF (seconds)	Speed-ups
AHB	2015/7/7	1230.1	38.7	31.8
	2015/9/25	1243.7	35.6	34.9
Tianjin	2018/10/1	775.5	18.3	42.5
	2018/12/4	742.9	16.9	44.1
Daxing	2018/4/8	535.1	12.3	43.4
	2018/10/1	531.1	11.9	44.7

1250 × 1250), in which case the computing time of serial procedures and data transmission between CPU and GPU only took negligible proportion of the total computing time. Considering the distinct variations in algorithm principle, we recorded the total running time of BiaSTF, FSDAF, and cuFSDAF using the AHB, Tianjin, and Daxing dataset (see Fig. 13). Because BiaSTF trained one model to predict two fusion images for each dataset, the total running time of BiaSTF includes the time of data processing, model training, and result predicting for two fusion images. For FSDAF and cuFSDAF, the total running time includes the running time for two fusion images. In our experiments, cuFSDAF was 1332.8, 1346.5, and 1359.9 times faster than BiaSTF for the three datasets, respectively. Considering cuFSDAF achieved similar accuracy and much better efficiency compared with FSDAF and BiaSTF, cuFSDAF is a better choice for large-scale and long-term spatiotemporal fusion tasks.

E. Computational Performance Using Various GPUs

To further assess the flexibility of cuFSDAF on GPUs with different computing capacities, we conducted a series of experiments using two other computers. The first computer was a computing node of the Tianhe-2 supercomputer, equipped with an Intel Xeon E5-2660 v3 CPU @ 2.6 GHz, and a Nvidia Tesla K80 GPU with 2,496 CUDA cores and 11,441 MB video memory. The second was a laptop computer with an Intel Core I5-7400 CPU @ 3.00 GHz, and Nvidia Geforce GTX 1050 GPU with 640 CUDA cores and 4 GB of video memory. The hardware and software environments of these two computers are shown in Table VII.

Compared with the C++-implemented FSDAF (hereafter called FSDAF), the cuFSDAF significantly improved the computational efficiency on both the Tianhe-2 supercomputer and the laptop computer Table VIII. Experimental results showed that cuFSDAF performed well in various hardware and software environments if a CUDA-enabled GPU was available. The higher the computing capacity of the GPU, the higher the performance of the cuFSDAF.

IV. CONCLUSION

Spatiotemporal data fusion algorithms are cost-efficient methods of obtaining remote sensing images with high spatial and temporal resolutions. Compared with other algorithms, the FSDAF algorithm closely captures the reflectance changes caused by land cover conversions and requires only one fine-resolution image as input. However, FSDAF faces several challenges when dealing with large-scale and long-term land surface dynamics because of its computational inefficiency. Moreover, block effects at the edges of subdomains are often inevitable in FSDAF fusion results. This study proposes an enhanced FSDAF algorithm parallelized using GPUs (cuFSDAF) to improve computational performance without losing accuracy.

The main enhancements of cuFSDAF include the following: 1) the TPS interpolator is replaced by an accelerated IDW interpolator to reduce computational complexity; 2) the algorithm is parallelized based on the CUDA, a widely used parallel computing framework for GPUs; and 3) an ADD method is proposed to improve the fusion accuracy at the edges of subdomains and to enable GPUs with different computing capacities to deal with datasets of any size.

The performance of cuFSDAF was evaluated through a series of experiments using the three sets of satellite images. Compared with FSDAF, cuFSDAF significantly improved the computational performance while maintaining accuracy. Also, cuFSDAF achieved similar accuracies compared to one of the latest deep-learning-based algorithms, BiaSTF, and much higher computational efficiency. cuFSDAF also performed better than popular algorithms such as STARFM and ESTRFM in areas with abrupt land cover changes in both accuracy and computational efficiency (results not shown). Such an improvement in computational efficiency greatly increases the feasibility and applicability of cuFSDAF to applications for large-scale long-term land surface dynamics and mass production. The code and test data of cuFSDAF are freely available at https://github.com/HPSCIL/cuFSDAF.

It is worth noting that besides IDW, other interpolators, such as bilinear and bicubic, can also be used to replace the TPS interpolator for FSDAF. In this study, IDW was used because it is easy to implement and parallelize, and the experiments showed that the accelerated IDW can achieve similar accuracy as TPS for FSDAF. Other interpolators can also be used if they can achieve equivalent or better accuracy than IDW.

The parallelization framework and strategies (e.g., ADD) proposed in our study not only can greatly increase the computational efficiency of FSDAF, but also can be used in other similar algorithms to enhance their computational efficiency and eventually improve their applicability in the long-term and large-scale fusion tasks. For example, the variants of FSDAF can easily adopt the proposed parallelization framework and strategies to achieve much higher efficiency because these FSDAF-like methods share similar principles and procedures. Moreover, other spatiotemporal fusion algorithms that include similar procedures/operations as FSDAF can also benefit from the proposed parallelization strategies. As a matter of fact, we have also parallelized three other spatiotemporal fusion algorithms, including STARFM [10], ESTARFM [13], and STNLFFM [18] (the source codes and test datasets are freely available at https://github.com/HPSCIL), which achieved significant improvements of computational efficiency on various GPUs.

Also, this study has several implications for the future development of spatiotemporal fusion. First, as shown in our experiments, the parallel procedures in cuFSDAF showed significant improvements in efficiency. Therefore, parallel computing on GPUs is a promising solution to enhance the efficiency of spatiotemporal fusion algorithms. Second, if the input images are decomposed before calculation, algorithms should take into account the multilevel neighborhood-scope calculations to avoid block effects at the edges of subdomains. Enlarging the size of subdomains and extending extra "halo" pixels should be considered. Third, for large-scale and long-term applications, algorithms should find a balance between accuracy and efficiency. When dealing with large-scale and long-term applications, the computational efficiency of the algorithm is as important as its accuracy. When enhancing existing algorithms, replacing computationally intensive procedures with more efficient procedures may be advisable as long as the variation in accuracy is acceptable.

Acknowledgment

The authors sincerely thank Dr. Jun Li at Sun Yat-sen University for kindly providing the software of BiaSTF.

References

[1] G. Xingfa and T. Xudong, "Overview of China Earth observation satellite programs [space agencies]," *IEEE Geosci. Remote Sens. Mag.*, vol. 3, no. 3, pp. 113–129, Sep. 2015.

[2] B. Zhukov, D. Oertel, F. Lanzl, and G. Reinhackel, "Unmixing-based multisensor multiresolution image fusion," *IEEE Trans. Geosci. Remote Sens.*, vol. 37, no. 3, pp. 1212–1226, May 1999.

[3] M.-F. Racault, S. Sathyendranath, and T. Platt, "Impact of missing data on the estimation of ecological indicators from satellite ocean-colour time-series," *Remote Sens. Environ.*, vol. 152, pp. 15–28, Sep. 2014.

[4] X. Zhu, F. Cai, J. Tian, and T. Williams, "Spatiotemporal fusion of multisource remote sensing data: Literature survey, taxonomy, principles, applications, and future directions," *Remote Sens.*, vol. 10, no. 4, p. 527, Mar. 2018.

[5] J. J. Settle and N. A. Drake, "Linear mixing and the estimation of ground cover proportions," *Int. J. Remote Sens.*, vol. 14, no. 6, pp. 1159–1177, Apr. 1993.

[6] R. Zurita-Milla, J. Clevers, and M. E. Schaepman, "Unmixing-based Landsat TM and MERIS FR data fusion," *IEEE Geosci. Remote Sens. Lett.*, vol. 5, no. 3, pp. 453–457, Jul. 2008.

[7] Z. Niu, "Use of MODIS and Landsat time series data to generate high-resolution temporal synthetic Landsat data using a spatial and temporal reflectance fusion model," *J. Appl. Remote Sens.*, vol. 6, no. 1, Mar. 2012, Art. no. 063507.

[8] J. Amorós-López *et al.*, "Multitemporal fusion of Landsat/TM and ENVISAT/MERIS for crop monitoring," *Int. J. Appl. Earth Observ. Geoinf.*, vol. 23, pp. 132–141, Aug. 2013.

[9] W. Zhang *et al.*, "An enhanced spatial and temporal data fusion model for fusing Landsat and MODIS surface reflectance to generate high temporal Landsat-like data," *Remote Sens.*, vol. 5, no. 10, pp. 5346–5368, Oct. 2013.

[10] F. Gao, J. Masek, M. Schwaller, and F. Hall, "On the blending of the Landsat and MODIS surface reflectance: Predicting daily Landsat surface reflectance," *IEEE Trans. Geosci. Remote Sens.*, vol. 44, no. 8, pp. 2207–2218, Aug. 2006.

[11] D. P. Roy *et al.*, "Multi-temporal MODIS–Landsat data fusion for relative radiometric normalization, gap filling, and prediction of Landsat data," *Remote Sens. Environ.*, vol. 112, no. 6, pp. 3112–3130, Jun. 2008.

[12] T. Hilker *et al.*, "A new data fusion model for high spatial- and temporal-resolution mapping of forest disturbance based on Landsat and MODIS," *Remote Sens. Environ.*, vol. 113, no. 8, pp. 1613–1627, Aug. 2009.

[13] X. Zhu, J. Chen, F. Gao, X. Chen, and J. G. Masek, "An enhanced spatial and temporal adaptive reflectance fusion model for complex heterogeneous regions," *Remote Sens. Environ.*, vol. 114, no. 11, pp. 2610–2623, Nov. 2010.

[14] T. Hwang, C. Song, P. V. Bolstad, and L. E. Band, "Downscaling real-time vegetation dynamics by fusing multi-temporal MODIS and Landsat NDVI in topographically complex terrain," *Remote Sens. Environ.*, vol. 115, no. 10, pp. 2499–2512, Oct. 2011.

[15] Q. Weng, P. Fu, and F. Gao, "Generating daily land surface temperature at Landsat resolution by fusing Landsat and MODIS data," *Remote Sens. Environ.*, vol. 145, pp. 55–67, Apr. 2014.

[16] P. Wu, H. Shen, L. Zhang, and F.-M. Göttsche, "Integrated fusion of multi-scale polar-orbiting and geostationary satellite observations for the mapping of high spatial and temporal resolution land surface temperature," *Remote Sens. Environ.*, vol. 156, pp. 169–181, Jan. 2015.

[17] Q. Wang *et al.*, "Fusion of Landsat 8 OLI and sentinel-2 MSI data," *IEEE Trans. Geosci. Remote Sens.*, vol. 55, no. 7, pp. 3885–3899, Jul. 2017.

[18] Q. Cheng, H. Liu, H. Shen, P. Wu, and L. Zhang, "A spatial and temporal nonlocal filter-based data fusion method," *IEEE Trans. Geosci. Remote Sens.*, vol. 55, no. 8, pp. 4476–4488, Aug. 2017.

[19] M. Belgiu and A. Stein, "Spatiotemporal image fusion in remote sensing," *Remote Sens.*, vol. 11, no. 7, p. 818, Apr. 2019.

[20] A. Li, Y. Bo, Y. Zhu, P. Guo, J. Bi, and Y. He, "Blending multi-resolution satellite sea surface temperature (SST) products using Bayesian maximum entropy method," *Remote Sens. Environ.*, vol. 135, pp. 52–63, Aug. 2013.

[21] H. Shen, X. Meng, and L. Zhang, "An integrated framework for the spatio–temporal–spectral fusion of remote sensing images," *IEEE Trans. Geosci. Remote Sens.*, vol. 54, no. 12, pp. 7135–7148, Dec. 2016.

[22] J. Xue, Y. Leung, and T. Fung, "A Bayesian data fusion approach to spatio-temporal fusion of remotely sensed images," *Remote Sens.*, vol. 9, no. 12, p. 1310, Dec. 2017.

[23] Y. Ke, J. Im, S. Park, and H. Gong, "Downscaling of MODIS one kilometer evapotranspiration using Landsat-8 data and machine learning approaches," *Remote Sens.*, vol. 8, no. 3, p. 215, Mar. 2016.

[24] B. Huang and H. Song, "Spatiotemporal reflectance fusion via sparse representation," *IEEE Trans. Geosci. Remote Sens.*, vol. 50, no. 10, pp. 3707–3716, Oct. 2012.

[25] B. Huang, H. Zhang, H. Song, J. Wang, and C. Song, "Unified fusion of remote-sensing imagery: Generating simultaneously high-resolution synthetic spatial–temporal–spectral Earth observations," *Remote Sens. Lett.*, vol. 4, no. 6, pp. 561–569, Jun. 2013.

[26] H. Song and B. Huang, "Spatiotemporal satellite image fusion through one-pair image learning," *IEEE Trans. Geosci. Remote Sens.*, vol. 51, no. 4, pp. 1883–1896, Apr. 2013.

[27] B. Wu, B. Huang, and L. Zhang, "An error-bound-regularized sparse coding for spatiotemporal reflectance fusion," *IEEE Trans. Geosci. Remote Sens.*, vol. 53, no. 12, pp. 6791–6803, Dec. 2015.

[28] V. Moosavi, A. Talebi, M. H. Mokhtari, S. R. F. Shamsi, and Y. Niazi, "A wavelet-artificial intelligence fusion approach (WAIFA) for blending Landsat and MODIS surface temperature," *Remote Sens. Environ.*, vol. 169, pp. 243–254, Nov. 2015.

[29] H. Song, Q. Liu, G. Wang, R. Hang, and B. Huang, "Spatiotemporal satellite image fusion using deep convolutional neural networks," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 11, no. 3, pp. 821–829, Mar. 2018.

[30] Y. Li, J. Li, L. He, J. Chen, and A. Plaza, "A new sensor bias-driven spatio-temporal fusion model based on convolutional neural networks," *Sci. China Inf. Sci.*, vol. 63, no. 4, Apr. 2020, Art. no. 140302.

[31] X. Liu, C. Deng, S. Wang, G.-B. Huang, B. Zhao, and P. Lauren, "Fast and accurate spatiotemporal fusion based upon extreme learning machine," *IEEE Geosci. Remote Sens. Lett.*, vol. 13, no. 12, pp. 2039–2043, Dec. 2016.

[32] C. M. Gevaert and F. J. García-Haro, "A comparison of STARFM and an unmixing-based algorithm for Landsat and MODIS data fusion," *Remote Sens. Environ.*, vol. 156, pp. 34–44, Jan. 2015.

[33] Y. Rao, X. Zhu, J. Chen, and J. Wang, "An improved method for producing high spatial-resolution NDVI time series datasets with multi-temporal MODIS NDVI data and Landsat TM/ETM+ images," *Remote Sens.*, vol. 7, no. 6, pp. 7865–7891, Jun. 2015.

[34] X. Zhu, E. H. Helmer, F. Gao, D. Liu, J. Chen, and M. A. Lefsky, "A flexible spatiotemporal method for fusing satellite images with different resolutions," *Remote Sens. Environ.*, vol. 172, pp. 165–177, Jan. 2016.

[35] J. Quan, W. Zhan, T. Ma, Y. Du, Z. Guo, and B. Qin, "An integrated model for generating hourly Landsat-like land surface temperatures over heterogeneous landscapes," *Remote Sens. Environ.*, vol. 206, pp. 403–423, Mar. 2018.

[36] Q. Wang and P. M. Atkinson, "Spatio-temporal fusion for daily sentinel-2 images," *Remote Sens. Environ.*, vol. 204, pp. 31–42, Jan. 2018.

[37] Q. Wang, Y. Tang, X. Tong, and P. M. Atkinson, "Virtual image pair-based spatio-temporal fusion," *Remote Sens. Environ.*, vol. 249, Nov. 2020, Art. no. 112009.

[38] L. Zhang, Q. Weng, and Z. Shao, "An evaluation of monthly impervious surface dynamics by fusing Landsat and MODIS time series in the Pearl River Delta, China, from 2000 to 2015," *Remote Sens. Environ.*, vol. 201, pp. 99–114, Nov. 2017.

[39] Y. Cai, S. Liu, and H. Lin, "Monitoring the vegetation dynamics in the dongting lake wetland from 2000 to 2019 using the BEAST algorithm based on dense Landsat time series," *Appl. Sci.*, vol. 10, no. 12, p. 4209, Jun. 2020.

[40] H. Yang *et al.*, "Measuring the urban land surface temperature variations under Zhengzhou city expansion using Landsat-like data," *Remote Sens.*, vol. 12, no. 5, p. 801, Mar. 2020.

[41] J. Zhou *et al.*, "Sensitivity of six typical spatiotemporal fusion methods to different influential factors: A comparative study for a normalized difference vegetation index time series reconstruction," *Remote Sens. Environ.*, vol. 252, Jan. 2021, Art. no. 112130.

[42] M. Liu *et al.*, "An improved flexible spatiotemporal DAta fusion (IFSDAF) method for producing high spatiotemporal resolution normalized difference vegetation index time series," *Remote Sens. Environ.*, vol. 227, pp. 74–89, Jun. 2019.

[43] X. Xie and A. Li, "Development of a topographic-corrected temperature and greenness model (TG) for improving GPP estimation over mountainous areas," *Agricult. Forest Meteorol.*, vol. 295, Dec. 2020, Art. no. 108193.

[44] R. Li *et al.*, "Phenology-based classification of crop species and rotation types using fused MODIS and Landsat data: The comparison of a random-forest-based model and a decision-rule-based model," *Soil Tillage Res.*, vol. 206, Feb. 2021, Art. no. 104838.

[45] Shi *et al.*, "A comprehensive and automated fusion method: The enhanced flexible spatiotemporal DAta fusion model for monitoring dynamic changes of land surface," *Appl. Sci.*, vol. 9, no. 18, p. 3693, Sep. 2019.

[46] X. Li *et al.*, "SFSDAF: An enhanced FSDAF that incorporates sub-pixel class fraction change information for spatio-temporal image fusion," *Remote Sens. Environ.*, vol. 237, Feb. 2020, Art. no. 111537.

[47] D. Guo, W. Shi, M. Hao, and X. Zhu, "FSDAF 2.0: Improving the performance of retrieving land cover changes and preserving spatial details," *Remote Sens. Environ.*, vol. 248, Oct. 2020, Art. no. 111973.

[48] P. Li *et al.*, "Human impact on suspended particulate matter in the Yellow River Estuary, China: Evidence from remote sensing data fusion using an improved spatiotemporal fusion method," *Sci. Total Environ.*, vol. 750, Jan. 2021, Art. no. 141612.

[49] Q. Guan and K. C. Clarke, "A general-purpose parallel raster processing programming library test application using a geographic cellular automata model," *Int. J. Geographical Inf. Sci.*, vol. 24, no. 5, pp. 695–722, Apr. 2010.

[50] C. Yang *et al.*, "Contemporary computing technologies for processing big spatiotemporal data," in *Space-Time Integration in Geography GIScience*, M.-P. Kwan, D. Richardson, D. Wang, C. Zhou, Eds. Dordrecht, The Netherlands: Springer, 2015, pp. 327–351.

[51] B. Zhang *et al.*, "Remotely sensed big data: Evolution in model development for information extraction [point of view]," *Proc. IEEE*, vol. 107, no. 12, pp. 2294–2301, Dec. 2019.

[52] J. Chen *et al.*, "Global land cover mapping at 30 m resolution: A POK-based operational approach," *ISPRS J. Photogramm. Remote Sens.*, vol. 103, pp. 7–27, May 2015.

[53] M. Schmitt, L. H. Hughes, C. Qiu, and X. X. Zhu, "SEN12MS—A curated dataset of georeferenced multi-spectral sentinel-1/2 imagery for deep learning and data fusion," *ISPRS Ann. Photogramm., Remote Sens. Spatial Inf. Sci.*, vol. IV-2/W7, pp. 153–160, Sep. 2019.

[54] P. Gong *et al.*, "Stable classification with limited sample: Transferring a 30-m resolution sample set collected in 2015 to mapping 10-m resolution global land cover in 2017," *Sci. Bull.*, vol. 64, no. 6, pp. 370–373, Mar. 2019.

[55] B. Li, H. Zhao, and Z. Lv, "Parallel ISODATA clustering of remote sensing images based on MapReduce," in *Proc. Int. Conf. Cyber-Enabled Distrib. Comput. Knowl. Discovery*, Huangshan, China, Oct. 2010, pp. 380–383.

[56] Z. Lv, Y. Hu, H. Zhong, J. Wu, B. Li, and H. Zhao, "Parallel k-means clustering of remote sensing images based on MapReduce," in *Web Information Systems and Mining*, vol. 6318, F. L. Wang, Z. Gong, X. Luo, and J. Lei, Eds. Berlin, Germany: Springer, 2010, pp. 162–170.

[57] Q. Guan, P. C. Kyriakidis, and M. F. Goodchild, "A parallel computing approach to fast geostatistical areal interpolation," *Int. J. Geographical Inf. Sci.*, vol. 25, no. 8, pp. 1241–1267, Aug. 2011.

[58] X. Shi and F. Ye, "Kriging interpolation over heterogeneous computer architectures and systems," *GISci. Remote Sens.*, vol. 50, no. 2, pp. 196–211, Apr. 2013.

[59] H. Zhu, Y. Cao, Z. Zhou, and M. Gong, "Parallel multi-temporal remote sensing image change detection on GPU," in *Proc. IEEE 26th Int. Parallel Distrib. Process. Symp. Workshops PhD Forum*, Shanghai, China, May 2012, pp. 1898–1904.

[60] C. Chen, Z. Chen, M. Li, Y. Liu, L. Cheng, and Y. Ren, "Parallel relative radiometric normalisation for remote sensing image mosaics," *Comput. Geosci.*, vol. 73, pp. 28–36, Dec. 2014.

[61] B. C. Pijanowski, A. Tayyebi, J. Doucette, B. K. Pekin, D. Braun, and J. Plourde, "A big data urban growth simulation at a national scale: Configuring the GIS and neural network based land transformation model to run in a high performance computing (HPC) environment," *Environ. Model. Softw.*, vol. 51, pp. 250–268, Jan. 2014.

[62] Q. Guan, X. Shi, M. Huang, and C. Lai, "A hybrid parallel cellular automata model for urban growth simulation over GPU/CPU heterogeneous architectures," *Int. J. Geographical Inf. Sci.*, vol. 30, no. 3, pp. 494–514, Mar. 2016.

[63] J. Lopez-Fandino, P. Quesada-Barriuso, D. B. Heras, and F. Arguello, "Efficient ELM-based techniques for the classification of hyperspectral remote sensing images on commodity GPUs," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 8, no. 6, pp. 2884–2893, Jun. 2015.

[64] L.-J. Zhu, J. Liu, C.-Z. Qin, and A.-X. Zhu, "A modular and parallelized watershed modeling framework," *Environ. Model. Softw.*, vol. 122, Dec. 2019, Art. no. 104526.

[65] J. Fung and S. Mann, "Computer vision signal processing on graphics processing units," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, Montreal, QC, Canada, vol. 5, May 2004, pp. 5–93.

[66] J. D. Owens, M. Houston, D. Luebke, S. Green, J. E. Stone, and J. C. Phillips, "GPU computing," *Proc. IEEE*, vol. 96, no. 5, pp. 879–899, May 2008.

[67] J. Nickolls and W. J. Dally, "The GPU computing era," *IEEE Micro*, vol. 30, no. 2, pp. 56–69, Mar. 2010.

[68] M. E. Paoletti, J. M. Haut, X. Tao, J. P. Miguel, and A. Plaza, "A new GPU implementation of support vector machines for fast hyperspectral image classification," *Remote Sens.*, vol. 12, no. 8, p. 1257, Apr. 2020.

[69] J. A. G. Jaramago, M. E. Paoletti, J. M. Haut, R. Fernandez-Beltran, A. Plaza, and J. Plaza, "GPU parallel implementation of dual-depth sparse probabilistic latent semantic analysis for hyperspectral unmixing," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 12, no. 9, pp. 3156–3167, Sep. 2019.

[70] S. Bernabe, C. Garcia, F. D. Igual, G. Botella, M. Prieto-Matias, and A. Plaza, "Portability study of an OpenCL algorithm for automatic target detection in hyperspectral images," *IEEE Trans. Geosci. Remote Sens.*, vol. 57, no. 11, pp. 9499–9511, Nov. 2019.

[71] S. Bernabe, G. Martin, J. M. P. Nascimento, J. M. Bioucas-Dias, A. Plaza, and V. Silva, "Parallel hyperspectral coded aperture for compressive sensing on GPUs," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 9, no. 2, pp. 932–944, Feb. 2016.

[72] O. Dubrule, "Comparing splines and kriging," *Comput. Geosci.*, vol. 10, nos. 2–3, pp. 327–338, Jan. 1984.

[73] G. Donato and S. Belongie, "Approximate thin plate spline mappings," in *Proc. Eur. Conf. Comput. Vis.*, 2002, pp. 21–31.

[74] P. A. Hancock and M. F. Hutchinson, "Spatial interpolation of large climate data sets using bivariate thin plate smoothing splines," *Environ. Model. Softw.*, vol. 21, no. 12, pp. 1684–1694, Dec. 2006.

[75] G. Y. Lu and D. W. Wong, "An adaptive inverse-distance weighting spatial interpolation technique," *Comput. Geosci.*, vol. 34, no. 9, pp. 1044–1055, Sep. 2008.

[76] G. Sharma, A. Agarwala, and B. Bhattacharya, "A fast parallel gauss jordan algorithm for matrix inversion using CUDA," *Comput. Struct.*, vol. 128, pp. 31–37, Nov. 2013.

[77] J. Sanders and E. Kandrot, *CUDA by Example: An Introduction to General-Purpose GPU Programming, Portable Documents*. Reading, MA, USA: Addison-Wesley, 2010.

[78] S. Cook, *CUDA Programming: A Developer's Guide to Parallel Computing With GPUs*. Waltham, MA, USA: Morgan Kaufmann, 2012.

[79] L. Rodrigues, D. Leandro Borges, and L. Marcos Gonalves, "A locally adaptive edge-preserving algorithm for image interpolation," in *Proc. 15th Brazilian Symp. Comput. Graph. Image Process.*, Fortaleza-CE, Brazil, 2002, pp. 300–305.

[80] J. Li, Y. Li, L. He, J. Chen, and A. Plaza, "Spatio-temporal fusion for remote sensing data: An overview and new benchmark," *Sci. China Inf. Sci.*, vol. 63, no. 4, Apr. 2020, Art. no. 140301.

[81] G. H. Ball and D. J. Hall, "ISODATA, a novel method of data analysis and pattern classification," Stanford Res. Inst., Menlo Park, CA, USA, Tech. Rep. NTIS Report AD 699616, 1965.

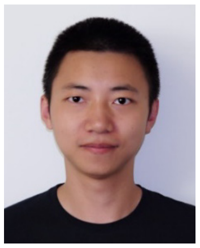
[82] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Trans. Image Process.*, vol. 13, no. 4, pp. 600–612, Apr. 2004.

[83] F. A. Kruse *et al.*, "The spectral image processing system (SIPS)-interactive visualization and analysis of imaging spectrometer data," in *Proc. AIP Conf.*, Pasadena, CA, USA, vol. 283, 1993, pp. 192–201.

[84] D. Renza, E. Martinez, and A. Arquero, "A new approach to change detection in multispectral images by means of ERGAS index," *IEEE Geosci. Remote Sens. Lett.*, vol. 10, no. 1, pp. 76–80, Jan. 2013.

Huan Gao received the B.E. degree in exploration engineering of mineral resources from the China University of Geosciences, Wuhan, China, in 2018, where he is pursuing the Ph.D. degree in surveying and mapping.

His research focuses on high-performance geospatial computing.

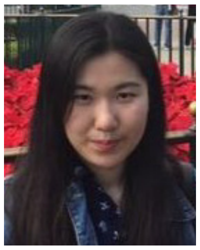


Xiaolin Zhu received the B.Sc. degree in resource science and engineering and the M.Sc. degree in civil engineering from Beijing Normal University, Beijing, China, in 2007 and 2010, respectively, and the Ph.D. degree in geography from The Ohio State University, Columbus, OH, USA, in 2014.

He was a Post-Doctoral Researcher with Colorado State University, Fort Collins, CO, USA, and the University of California at Davis, Davis, CA, USA, in 2015 and 2016, respectively. He is an Assistant Professor with the Department of Land Surveying and Geo-Informatics, The Hong Kong Polytechnic University, Hong Kong. His research interests include remote sensing image processing, data fusion, vegetation remote sensing, nighttime light remote sensing, and urban remote sensing.

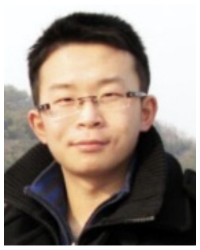
Qingfeng Guan received the B.S. degree in geography from East China Normal University, Shanghai, China, in 2000, the M.S. degree in geography from the Chinese Academy of Sciences, Beijing, China, in 2003, and the Ph.D. degree in geography from the University of California at Santa Barbara, Santa Barbara, CA, USA, in 2008.

He is a Professor with the School of Geography and Information Engineering, China University of Geosciences, Wuhan, China. His research interests include high-performance geospatial computing, spatial computational intelligence, and spatiotemporal modeling.



Xue Yang received the B.E. degree in surveying engineering from the China University of Geosciences, Wuhan, China, in 2018, where she is pursuing the Ph.D. degree with the School of Geography and Information Engineering.

Her research focuses on high-performance geospatial computing.

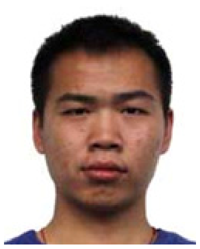


Yao Yao received the B.S. degree in surveying and mapping engineering and the M.S. degree in geodesy and surveying engineering from Wuhan University, Wuhan, China, in 2008 and 2011, respectively, and the Ph.D. degree in cartography and geographic information system from Sun Yat-sen University, Guangzhou, China, in 2017.

He is an Associate Professor with the School of Geography and Information Engineering, China University of Geosciences, Wuhan. His research interests include the application of geospatial big data and urban computing.

Wen Zeng received the B.S. degree in computer software from the Huazhong University of Science and Technology, Wuhan, China, in 1991, the M.S. degree in computer software from Jinan University, Guangzhou, China, in 1994, and the Ph.D. degree in cartography and geographic information engineering from the China University of Geosciences, Wuhan, in 2003.

He has been a Professor with the School of Geography and Information Engineering, China University of Geosciences, since 2007. His research interests include data acquisition and modeling for urban geographic information systems, geographic networks optimization, geographic information systems for transportation, and municipal infrastructure management. He has published more than 40 research articles, including peer-reviewed articles in international journals, such as *International Journal of Geographical Information Science*, *Annals of the American Association of Geographers*, and *Cities*.



Xuantong Peng received the B.E. degree in geographic information system and the M.S. degree in cartography and geographical information engineering from the China University of Geosciences, Wuhan, China, in 2016 and 2019, respectively.

His research focuses on high-performance geospatial computing.