Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2013, Article ID 405372, 10 pages
http://dx.doi.org/10.1155/2013/405372

Research Article

A New Tool for Intelligent Parallel Processing of Radar/SAR Remotely Sensed Imagery

A. Castillo Atoche,[1] R. Carrasco Alvarez,[2] J. Ortegón Aguilar,[3] and J. Vázquez Castillo[3]

[1] *Universidad Autónoma de Yucatán, Avenida Industrias No Contaminantes S/N, Apartado Postal 150, 97310 Mérida, YUC, Mexico*
[2] *Universidad de Guadalajara, Boulevard Marcelino García Barragán1421, 44430 Guadalajara, JAL, Mexico*
[3] *Universidad de Quintana Roo, Boulevard Bahía S/N Esquina Ignacio Comonfort, 77019 Chetumal, QROO, Mexico*

Correspondence should be addressed to J. Ortegón Aguilar; jortegon@uqroo.mx

Received 19 July 2013; Accepted 11 September 2013

Academic Editor: Marco Pérez-Cisneros

A novel parallel tool for large-scale image enhancement/reconstruction and postprocessing of radar/SAR sensor systems is addressed. The proposed parallel tool performs the following intelligent processing steps: image formation, for the application of different system-level effects of image degradation with a particular remote sensing (RS) system and simulation of random noising effects, enhancement/reconstruction by employing nonparametric robust high-resolution techniques, and image postprocessing using the fuzzy anisotropic diffusion technique which incorporates a better edge-preserving noise removal effect and faster diffusion process. This innovative tool allows the processing of high-resolution images provided with different radar/SAR sensor systems as required by RS endusers for environmental monitoring, risk prevention, and resource management. To verify the performance implementation of the proposed parallel framework, the processing steps are developed and specifically tested on graphic processing units (GPU), achieving considerable speedups compared to the serial version of the same techniques implemented in C language.

1. Introduction

The amount of data acquired by imaging satellites has been growing steadily in recent years. Many techniques of parallel computing and distributed systems are used by such imaging systems in novel remote sensing (RS) applications, which require timely responses for swift decisions. Relevant examples include monitoring of natural disasters like earthquakes and floods, military applications, tracking of man-induced hazards, forest fires, oil spills, and other types of biological agents. In addition, the acquisition of large-scale RS images with radar/SAR systems collects huge data of information [1]. This amount of data requires significant high computationally resources for applying image processing techniques in order to improve the quality of the images. Therefore, one solution to achieve the required computationally demanded issue goes through parallel approaches in a high performance computing (HPC) sense using PC clusters, grids, and clouds, among others [2]. In this regard, significant efforts have been realized to develop parallel tools for processing remotely sensed data [3–5]. In [3], Bernabé et al. present a classification for remotely sensing imaginary tool which implements the well-known unsupervised k-means and the ISODATA clustering techniques with Graphic Processing Units (GPU). However, this tool is focused only on classification issues. Another RS tool, developed by Shkvarko et al. [4], presents a simulation software for intelligent postprocessing of large scale RS imaginary. However, only computer simulations of the enhancement/reconstruction regularization of RS images, without parallel implementations, were carried out. On the other hand, an end-to-end image simulation tool for space and airborne imaging systems, called PICASO, was developed in [5]. This tool seeks to optimize image quality within real world constrains, but the algorithms are not implemented in a parallel scheme.

In this paper, a new parallel tool for processing remotely sensed images is addressed. The proposed parallel tool performs the following processing steps: image enhancement/reconstruction and postprocessing. First, the image formation technique applies different system-level effects

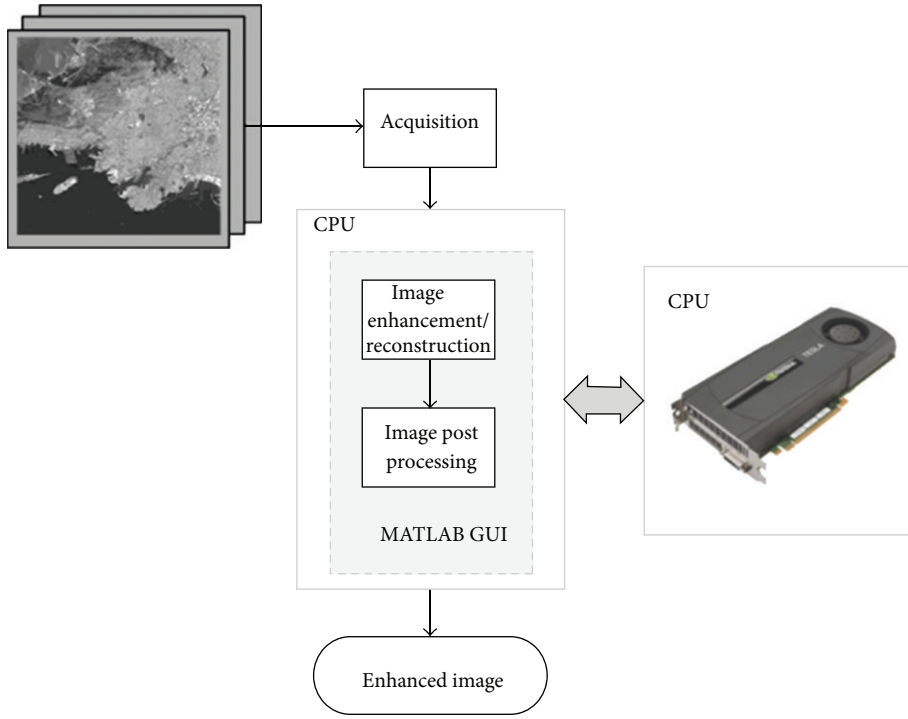


Figure 1: RS problem model block diagram.

in order to degrade the images (i.e., along the range and azimuth directions) and adding random noising effects. Second, the high-resolution enhancement/reconstruction of the power spatial spectrum pattern (SSP) of the wave field scattered from the extended remotely sensed scene is employed via the descriptive regularization approach with the Robust Space Filter (RSF) and the Robust Adaptive Space Filter (RASF) algorithms [6]. Third, the fuzzy anisotropic diffusion post-processing technique is applied. This last step is performed via the modification of the well-known Perona-Malik (PM) anisotropic diffusion technique [7] via the incorporation of fuzzy logic sets. The *model-free* fuzzy anisotropic diffusion post-processing technique, the "*model-based*" enhancement/reconstruction regularization algorithms, and the image formation step are unified into a new tool for providing intelligent processing of RS imagery.

The set of steps described above will provide to the end user a tool with the aim of improving the quality of the RS images acquired from radar/SAR systems; however, there is an important drawback for its high computational cost. Therefore, the proposed parallel tool is developed using GPU. Nowadays, these specialized hardware devices have evolved into a highly parallel, multithreaded, many-core processors with tremendous computational speed and very high memory bandwidth [2]. As a result, the proposed framework provides very powerful data processing and analysis capabilities coupled to the parallel computing and data resources in a manner that is transparent to the user.

The rest of the paper is organized as follows. In Section 2, a brief summary of the intelligent data processing steps employed in the proposed system is presented. The GPU-based data processing techniques for the parallel algorithm implementation is performed and detailed in Section 3. In Section 4, the results of the proposed approach and the performance analysis with the GPU platform are presented. The application of two case studies for the intelligent data processing of large-scale images with the proposed parallel tool is included in order to verify the accuracy and the time performance of the system. Finally, the concluding remarks are presented in Section 5.

2. Intelligent Data Processing Design Steps for Remote Sensing Imaging Problems

In this section, we present a brief overview of the general processing chain employed in this study. The addressed methodology for real-time formation/enhancement/reconstruction/post-processing of the RS imagery acquired with radar/SAR systems is described. Figure 1 presents the design flow methodology considered in this work.

2.1. Image Formation. In this section, we present the summary of the RS imaging problem that was previously developed in [6, 8]. Let us consider the measurement data wave field $u(\mathbf{y}) = F(\mathbf{y}) + n(\mathbf{y})$ modeled as a superposition of the echo signals F and additive noise n assumed to be available for observations and recordings within the prescribed time-space observation domain $Y \ni \mathbf{y}$, where $\mathbf{y} = (t, \mathbf{p})^T$ defines the time-space points in the observation domain $Y = T \times P$. The model of observation wave field u is specified

by the linear stochastic equation of observation (EO) with operator form [9]: $u = Fe + n$; $e \in E$; $u, n \in U$; $F : E \to U$. Next, we take into account the conventional finite-dimensional vector form approximation [6, 8, 10] of the continuous-form EO, where the data acquisition model is defined by a set of equations as

$$\mathbf{u}^{(m)} = \mathbf{F}^{(m)}\mathbf{e} + \mathbf{n}^{(m)}, \tag{1}$$

for M methods/systems to be aggregated/fused, that is, $m = 1, \ldots, M$, where $\mathbf{F}^{(m)}$ represent the system/method degradation operators usually referred to as the imaging system point spread functions (PSF), and vector $\mathbf{n}^{(m)}$ represents the noise in the actually acquired image, respectively. The mean $\mathbf{b} = \text{vect}\{\langle e_k, e_k^* \rangle;\ k = 1, \ldots, K\}$ of the random scattering vector $\mathbf{e}$ has a statistical meaning of the average power scattering function traditionally referred as the spatial spectrum pattern (SSP), where the asterisk indicates the complex conjugate. This SSP is a second-order statistics of the scattered field that represent the brightness reflectivity of the image scene, represented in a conventional pixel format over the rectangular scene frame [9].

2.2. Image Enhancement/Reconstruction. In this stage, we estimate $\widehat{\mathbf{b}}$ as a discrete-form representation of the desired SSP over the pixel-formatted object scene remotely sensed with an employed array radar/SAR. Thus, one can seek to estimate $\widehat{\mathbf{b}} = \{\widehat{\mathbf{R}}_{\mathbf{e}}\}_{\text{diag}}$ given the data correlation matrix $\mathbf{R}_{\mathbf{u}}$ preestimated by some means, for example, via averaging the correlations over L independent snapshots [11, 12]; $\widehat{\mathbf{R}}_{\mathbf{u}} = \mathbf{Y} = \text{aver}_{l \in L}\{\mathbf{u}_{(l)}\mathbf{u}_{(l)}^{+}\} = (l/L)\sum_{l=1}^{L}\mathbf{u}_{(l)}\mathbf{u}_{(l)}^{+}$, and by determining the solution operator that we also refer to as the signal image formation operator (SO) $\mathbf{G}$ such that

$$\widehat{\mathbf{b}} = \left\{\widehat{\mathbf{R}}_{\mathbf{e}}\right\}_{\text{diag}} = \left\{\mathbf{G}\mathbf{Y}\mathbf{G}^{+}\right\}_{\text{diag}}. \tag{2}$$

Such image enhancement/reconstruction processing tasks can be mathematically formalized in terms of the following optimization problem [9]:

$$\mathbf{G} = \arg\min_{\mathbf{G}}\left[\min_{\langle \|\Delta\|^2 \rangle_{p(\Delta)} \le \delta}\{\mathfrak{R}(\mathbf{G})\}\right], \tag{3}$$

in which $\mathfrak{R}(\mathbf{G}) = \text{tr}\{(\mathbf{GF}-\mathbf{I})\mathbf{A}(\mathbf{GF}-\mathbf{I})^{+}\} + \alpha\,\text{tr}\{\mathbf{G}\mathbf{R}_{\mathbf{u}}\mathbf{G}^{+}\}$ represents the objective function where the first term (systematic risk component) is performed over the randomness of the distorted SFO $\mathbf{G}$ with the uncertainty conditioned by the statistical bound $\langle \|\Delta\|^2 \rangle_{p(\Delta)} \le \delta$. The regularization parameter α balances the systematic risk component (specified by the first term) and the fluctuation risk component (specified by the second term) in $\mathfrak{R}(\mathbf{G})$, while $\mathbf{A}$ induces the weighted metrics structure in the systematic risk. The regularization parameter α and the invertible weight matrix $\mathbf{A}$ determine the user adjustable "degrees of freedom."

In this regard, the solution to the optimization problem derived in previous studies [8, 13] is described as follows:

$$\mathbf{G} = \mathbf{K}\mathbf{F}^{+}\mathbf{R}_{\mathbf{n}}^{-1}, \quad \text{where } \mathbf{K} = \left(\mathbf{F}^{+}\mathbf{F} + \alpha\mathbf{A}^{-1}\right)^{-1}\mathbf{F}^{+}\mathbf{R}_{\mathbf{n}}^{-1}, \tag{4}$$

where $\mathbf{K}$ defines the so-called reconstruction operator (with the regularization parameter α and stabilizer $\mathbf{A}^{-1}$), and $\mathbf{R}_{\mathbf{n}}^{-1}$ is the inverse of the diagonal loaded noise correlation matrix [14], that is, $\mathbf{R}_{\mathbf{n}} = N_0\mathbf{I}$, attributing the unknown correlated noise component. Thus, for different methods, different cost functions are employed. Here we exemplify two practically motivated high-resolution reconstructive imaging techniques [6, 15, 16] that will be used at the parallel toolbox: (i) the robust spatial filtering (RSF) and (ii) the robust adaptive spatial filtering (RASF) methods.

(i) *RSF.* The RSF method implies no preference to any prior model information (i.e., $\alpha\mathbf{A}^{-1} = \mathbf{I}$) and balanced minimization of the systematic and noise error measures by adjusting the regularization parameter α to the inverse of the signal-to-noise ratio (SNR). In that case the SO becomes the Tikhonov-type robust spatial filter (RSF) [6]: $\mathbf{G}_{\text{RSF}} = \mathbf{G}^{(1)} = \left(\mathbf{F}^{+}\mathbf{F} + \alpha\mathbf{A}^{-1}\right)^{-1}\mathbf{F}^{+}$, in which the RSF regularization parameter α_{RSF} is adjusted to a particular operational scenario model, namely, $\alpha_{\text{RSF}} = (N_0/b_0)$ for the case of a certain operational scenario, where N_0 represents the white observation noise power density and b_0 is the average a priori SSP value.

(ii) *RASF.* In this method, α and $\mathbf{A}$ are adjusted in an adaptive fashion following the minimum risk strategy [9], that is, $\alpha\mathbf{A}^{-1} = \widehat{\mathbf{D}} = \text{diag}(\widehat{\mathbf{b}})$, the diagonal matrix with the estimate $\widehat{\mathbf{b}}$ at its principal diagonal, in which case the SO becomes itself the solution-dependent operator that results in the following robust adaptive spatial filter (RASF): $\mathbf{G}_{\text{RASF}} = \mathbf{G}^{(2)} = (\mathbf{F}^{+}\mathbf{R}_{\mathbf{n}}^{-1}\mathbf{F} + \widehat{\mathbf{D}}^{-1})^{-1}\mathbf{F}^{+}\mathbf{R}_{\mathbf{n}}^{-1}$. In all practical RS scenarios (and, specifically, in SAR uncertain imaging applications [17–19]), it is a common practice to accept the robust white additive noise model, that is, $\mathbf{R}_{\mathbf{n}} = N_0\mathbf{I}$, attributing the unknown correlated noise component.

Any other feasible adjustments in the degrees of freedom (the regularization parameter α and the weight matrix $\mathbf{A}$) provide other possible SSP reconstruction techniques that we do not consider in this study.

2.3. Image Postprocessing. In this section, the aggregation of the fuzzy anisotropic diffusion and the regularization-based methods are described for the reconstruction/post-processing image processing. The authors consider that the aggregation of such methods in the proposed parallel toolbox increases the flexibility and robust edge definition in the reconstructed image. That is, the noise and the edges are both high frequency image components, and the conventional image enhancement/reconstruction techniques oriented for large-scale remote sensing imaging do not work well for edge-preserving smoothing of image corrupted with additive and speckle noise (i.e., a tradeoff between sharpening and blurring must be selected). In this study, the gradient as the edge factor in the anisotropic diffusion is replaced by a rule-based fuzzy anisotropic diffusion. Also, a fuzzy inference

system is employed to replace the edge stopping function providing a better control on the diffusion process.

The aggregation of the reconstruction/post-processing framework offers the possibility to preserve high spatial resolution performances via anisotropic diffusion image post-processing, which implies anisotropic regularizing windowing (WO) over the reconstructed solution in the image space. Next, the following equation represents the celebrated Perona-Malik anisotropic diffusion method [7]:

$$\frac{\partial \widehat{b}(\mathbf{r};t)}{\partial t} = \operatorname{div}\left(c(\mathbf{r};t)\,\nabla\widehat{b}(\mathbf{r};t)\right), \tag{5}$$

for the continuous 2D rectangular scene frame $R \ni \mathbf{r} = (x, y)$. In (5), $\partial\widehat{b}(\mathbf{r};t)/\partial t$ represents the evolutionary reconstructed SSP estimate $\widehat{b}(\mathbf{r};t)$, which provides edge preservation in the scene regions with high gradient contrast while performing smoothed windowing over the homogeneous image zones corrupted by speckle noise. Also, $c(\mathbf{r};t) = g(\|\nabla\widehat{b}(\mathbf{r};t)\|)$ represents the diffusion coefficient, $\nabla\widehat{b}(\mathbf{r};t)$ denotes the gradient of the image, and $g(\cdot)$ is called edge stopping function, which is selected as a decreasing function of the gradient of the reconstructed image.

Now, the discrete version of the anisotropic diffusion equation of (5) is represented as

$$\widehat{b}_{(r)}^{[i+1]} = \widehat{b}_{(r)}^{[i]} + \frac{\lambda}{|\eta|}\sum_{p\epsilon\eta} g\left(\nabla\widehat{b}_{(r,p)}\right)\nabla\widehat{b}_{(r,p)}, \tag{6}$$

where $|\eta|$ is equal to the number of pixels in the neighborhood that is normally at the boundaries in the north, south, east, and west directions. $\nabla\widehat{b}_{(r,p)}$ is the discrete gradient in one of four diffusion directions. The diffusion coefficient is next defined as follows:

$$g\left(\widehat{b}_r, k\right) = \frac{1}{1 + \left(\nabla\widehat{b}_{(r,p)}/K\right)^2}, \tag{7}$$

where parameter K is called the scale parameter, which should be selected to be smaller than the gradient at edges and larger than the gradient at noise. Furthermore, because of inherent noise, the calculated edge strength based on gradient is not a true reflection of the edge strength. Therefore, the ambiguity of choosing a suitable value for parameter K and thus the uncertainty in the diffusion coefficient justify the use of fuzzy set theory in such situations. The discrete version of the anisotropic diffusion equation of (6) is represented as

$$\begin{aligned}\widehat{b}_r^{[i+1]} = \widehat{b}_r^{[i]} &+ \lambda\left[\frac{1}{d_x^2}\cdot c_{\mathrm{N}}\cdot D_{\mathrm{N}}\left(\widehat{b}\right) + \frac{1}{d_x^2}\cdot c_{\mathrm{S}}\cdot D_{\mathrm{S}}\left(\widehat{b}\right)\right.\\ &+ \frac{1}{d_y^2}\cdot c_{\mathrm{E}}\cdot D_{\mathrm{E}}\left(\widehat{b}\right) + \frac{1}{d_y^2}\cdot c_{\mathrm{W}}\cdot D_{\mathrm{W}}\left(\widehat{b}\right)\\ &+ \frac{1}{d_d^2}\cdot c_{\mathrm{NE}}\cdot D_{\mathrm{NE}}\left(\widehat{b}\right) + \frac{1}{d_d^2}\cdot c_{\mathrm{NW}}\cdot D_{\mathrm{NW}}\left(\widehat{b}\right)\\ &\left.+ \frac{1}{d_d^2}\cdot c_{\mathrm{SE}}\cdot D_{\mathrm{SE}}\left(\widehat{b}\right) + \frac{1}{d_d^2}\cdot c_{\mathrm{SW}}\cdot D_{\mathrm{SW}}\left(\widehat{b}\right)\right]_r^{[i]}\end{aligned} \tag{8}$$

in which c_{N} through c_{SW} represents the diffusion coefficients, d_x, d_y, and d_d are the distance between the pixels, and D_{N} through D_{SW} indicate the nearest-neighbor differences for the corresponding direction.

In this study, we propose to change each nearest-neighbor differences $\nabla\widehat{b}_{(r,p)}$ (D in the discrete representation) that define the edge factor (algorithmically the same as the Laplacian filter in each direction) in the traditional Perona-Malik algorithm of (6) with a Fuzzy Logic System (FLS) approach that calculates the edges in order to avoid the noise of the image [20].

Fuzzy logic system (FLS) is a rule-based theory in which an input is first fuzzified (converted from a crisp number to a fuzzy set) and subsequently processed by an inference engine that retrieves knowledge in the form of fuzzy rules contained in a rule-base. The fuzzy sets computed by the fuzzy inference as the output of each rule are then composed and defuzzified (converted from a fuzzy set to a crisp number). The following fuzzy rules are defined in Table 1.

The proposed fuzzy anisotropic diffusion approach is next described in detail as follows:

$$\begin{aligned}\widehat{b}_r^{[i+1]} = \widehat{b}_r^{[i]} &+ \lambda\left[\frac{1}{d_x^2}\cdot c_{\mathrm{N}}\cdot \Upsilon_{\mathrm{N}}\left(\widehat{b}\right) + \frac{1}{d_x^2}\cdot c_{\mathrm{S}}\cdot \Upsilon_{\mathrm{S}}\left(\widehat{b}\right)\right.\\ &+ \frac{1}{d_y^2}\cdot c_{\mathrm{E}}\cdot \Upsilon_{\mathrm{E}}\left(\widehat{b}\right) + \frac{1}{d_y^2}\cdot c_{\mathrm{W}}\cdot \Upsilon_{\mathrm{W}}\left(\widehat{b}\right)\\ &+ \frac{1}{d_d^2}\cdot c_{\mathrm{NE}}\cdot \Upsilon_{\mathrm{NE}}\left(\widehat{b}\right) + \frac{1}{d_d^2}\cdot c_{\mathrm{NW}}\cdot \Upsilon_{\mathrm{NW}}\left(\widehat{b}\right)\\ &\left.+ \frac{1}{d_d^2}\cdot c_{\mathrm{SE}}\cdot \Upsilon_{\mathrm{SE}}\left(\widehat{b}\right) + \frac{1}{d_d^2}\cdot c_{\mathrm{SW}}\cdot \Upsilon_{\mathrm{SW}}\left(\widehat{b}\right)\right]_r^{[i]},\end{aligned} \tag{9}$$

in which c_{N} through c_{SW} represent the diffusion coefficients computed using (7), d_x, d_y, and d_d are the distance between the pixels, and Υ_{N} through Υ_{SW} represent the edge in each direction as a result of the fuzzy logic system.

Now, we are ready to describe the GPU-based implementation of the intelligent parallel processing toolbox for remotely sensed imagery.

3. GPU-Based Parallel Toolbox Implementation

In order to speed up the processing design flow methodology described in the previous section, a parallel framework is used. This parallel tool was developed using the graphic

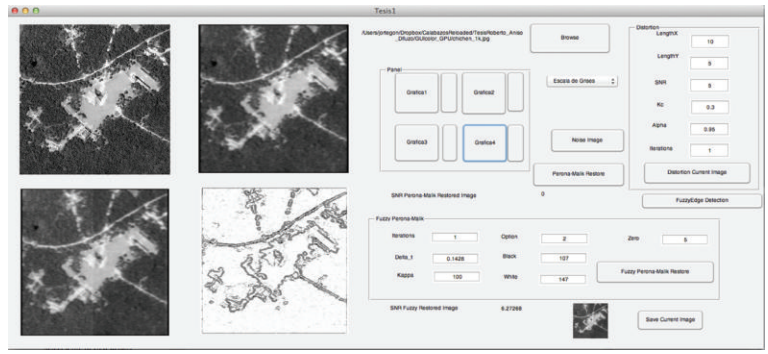


Figure 2: Graphic user interface.

Table 1: Fuzzy rules to compute F_N through F_{SW} for *Black* membership function.

Direction	Fuzzy rule
North	IF D_{NW} is *Zero* AND D_N is *Zero* AND D_{NE} is *Zero* THEN F_N is *Black* ELSE F_N is *White*
South	IF D_{SW} is *Zero* AND D_S is *Zero* AND D_{SE} is *Zero* THEN F_S is *Black* ELSE F_S is *White*
East	IF D_{NE} is *Zero* AND D_E is *Zero* AND D_{SE} is *Zero* THEN F_E is *Black* ELSE F_E is *White*
West	IF D_{NW} is *Zero* AND D_W is *Zero* AND D_{SW} is *Zero* THEN F_W is *Black* ELSE F_W is *White*
North east	IF D_N is *Zero* AND D_{NE} is *Zero* AND D_E is *Zero* THEN F_{NE} is *Black* ELSE F_{NE} is *White*
North west	IF D_N is *Zero* AND D_{NW} is *Zero* AND D_W is *Zero* THEN F_{NW} is *Black* ELSE F_{NW} is *White*
South east	IF D_E is *Zero* AND D_{SE} is *Zero* AND D_S is *Zero* THEN F_{SE} is *Black* ELSE F_{SE} is *White*
South west	IF D_W is *Zero* AND D_{SW} is *Zero* AND D_S is *Zero* THEN F_{SW} is *Black* ELSE F_{SW} is *White*

user interface (GUI) of MATLAB and the algorithmic implementation (i.e., image formation/reconstruction/post-processing) with GPU. The proposed approach also allows other functionalities, such as gray-scale image representation, random noising effects, image enhancement/reconstruction, fuzzy edge detection representation, and loading/storing of results for different radar/SAR systems. The developed parallel tool is shown in Figure 2. From the analysis of Figure 2, it is easy to observe that the toolbox presents the results for each corresponding processing stage.

In addition, the algorithmic implementation of the processing chain framework uses several processors working with independent data partitions [2]. However, before getting the details of the implementation let us remark that the implementation uses CUDA-enabled GPU, which are separate devices that are installed in a host computer, run asynchronously to the host processor, and have their own physical memory.

The basic unit of work on the GPU is a thread. Every thread acts as if it has its own processor with separate registers and identity that happens to run in a shared memory environment [2]. CUDA programs utilize kernels, which are subroutines callable from the host that work on the CUDA device. The extern function is called from the host, and it calls the different kernels. A kernel utilizes many threads to perform the work defined in the kernel source code. The kernel should make partitions of the data to be processed by each thread, taking care of not overlapping thread processing on any memory section in order to avoid undesired results.

The way the GPU is divided is in Streaming Multiprocessors (SMs), and each SM contains cores that execute and identical instruction set, or sleep; up to 32 threads may be scheduled at a time, called a warp, but maximum 24 warps are active in one SM. The threads in the same multiprocessor can share "Shared Memory" by synchronizing their execution for coordinating accesse to memory. The SM divides registers among threads and threads access the register memory as local; they have access to local cache memories in the multiprocessor, while the multiprocessors have access to the global GPU (device) memory.

The GPU blocks represent the number of parallel blocks that we will launch in our grid. Hardware is free to assign blocks to any processor at any time, and at execution time they tell GPU how many threads to launch per block [21]. Figure 3 shows how block and thread configuration would look in a small, 6 block, 100-thread kernel.

One of the first decisions to make before developing the GPU implementation is about the memory mapping of data, that is, how the GPU memory will be used for an improved performance. To do that, we assume that image fits into GPU memory, and in future works this problem will be addressed for larger images. However, instead of just copying and using the values for each pixel, we use textures. Textures are bound to global memory and can provide both cache and some limited, 9-bit processing capabilities [2]. The global memory is allocated as CUDA arrays; when the arrays are bound, textures are cache optimized for spatial locality and allow interpolation, wrapping, and clamping.

On the other hand, we use fuzzy rules which are common to each thread and do not change over time, and for them we use another kind of memory known as constant memory. Constant memory is used for data that will not change over the course of a kernel execution. In some situations, using constant memory rather than global memory will reduce the required memory bandwidth [21].

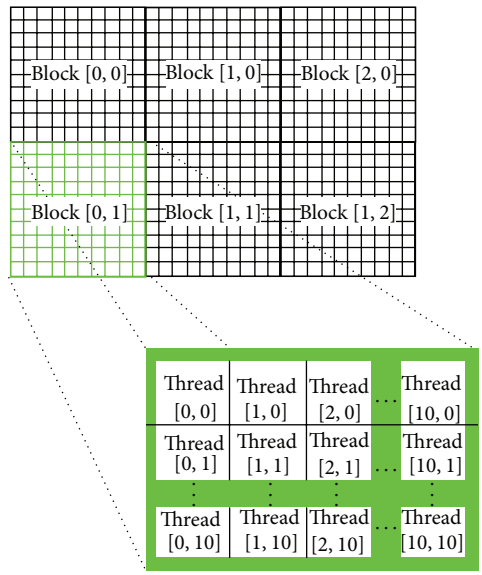


Figure 3: A 2D hierarchy of blocks and threads in an SM.

Considering the aggregation of parallel techniques with GPU computing, we next describe the efficient GPU-based implementation of each processing stage of the proposed parallel tool: GPU implementation of the (i) image formation; (ii) image enhancement/reconstruction; (iii) image post-processing.

Now, we describe the specific functions and CUDA kernels used for efficient implementation of image formation/reconstruction/post-processing on the GPU.

First, a kernel is implemented to compute the image formation processing applying the conventional Matched Space Filter (MSF) algorithm [22]. Due to the fact that this filter can be expressed as a Toeplitz matrix, the processing stage is highly improved due to its potential for parallelization in a massively parallel scheme. The *image-formation* kernel is launched with many threads as pixels contain the RS image, where each thread executes the corresponding operation of the MSF method. Additionally, the time processing of the algorithm is also reduced with the efficient management of data transfers from the CPU and GPU. The device overlap function was activated with the capacity to simultaneously execute a CUDA kernel while performing a copy between CPU to GPU memory [2, 21]. Multiple CUDA streams are created to perform this overlap of computation in data transfer.

Second, the computational procedures for the implementation of the RSF/RASF reconstructive algorithms are described. We start with the memory configuration management using the "mmap" library of GNU C. With this library, the image can be accessed from the CPU with a simple pointer without the necessity to read each pixel value. Next, the matrix-matrix multiplication operations are computed. A specific CUDA kernel is implemented using one grid of $n \times m$ blocks, in which each block processes the corresponding sub-matrix operation in a parallel scheme. The results are next loaded in the shared memory of the GPU. Note that several additional operations must be implemented in this stage. In this study, we propose to use the *reduction* algorithm [23].

Next, the extern function Perona-Malik is called from the host, it binds the texture to array, calls the "fuzzy" kernel for image enhancement, and normalizes the resulting array using the optimized NVIDIA Performance Primitives [24]. The Perona fuzzy kernel is computed for the image enhancement post-processing. In this stage, the first operation is to read the texture at subpixel precision and take differences in a neighbourhood window of 3×3 pixels; this corresponds to D_{N} through D_{SW} values for the fuzzy rules of Table 1. The fuzzy input vectors are used as membership function, and that this means the *Zero*, *Black*, and *White* values are used in the fuzzy rules as depicted in Table 1. We have 2 membership functions defined for each of the 8 possible directions (neighbours) of a pixel, resulting in 16 fuzzy values.

After that, it is applied the implication method (min function), afterwards, an aggregation method (max function), and finally defuzze the values applying the anisotropic diffusion.

The implication method is computed Imp_Y as the minimum between the resulting F_X and the fuzzy input value for *White* or *Black*, depending on the membership function used, where X is equal to N through SW, and Y is equal to White and Black. The aggregation method takes the maximum between $\mathrm{Imp}_{\mathrm{White}}$ and $\mathrm{Imp}_{\mathrm{Black}}$. Finally the anisotropic diffusion was applied using (9).

The resulting pixel values are independent from others, and hence the fuzzy calculations are suitable for a GPU implementation. Each thread is responsible for computing the fuzzy anisotropic diffusion of a part of the image. This image part is selected as shown in Figure 3; that is, it is done considering the block's size, block's index, and thread's index.

The resulting pixel values are float numbers; hence, in order to ease the visualization, they are normalized to unsigned char values for each channel. This task is performed before displaying the resulting image but is optional in case of post-processing. We take advantage of optimized functions provided by the Nvidia Performance Primitives [24], using the *nppsMinMax_32f* to get the minimum and maximum float value resulting from the fuzzy kernel and *nppsNormalize_32f* to normalize the image with the resulting minimum value at 0 and maximum at 255.

4. Results and Performance Analysis

In this section, we carry out the experimental validation of the proposed parallel toolbox system. Also, the time performance analysis is performed in order to demonstrate the time processing improvement achieved with the GPU-based implementation. The experimental case studies are next described for a high-resolution RS image acquired from an SAR system.

4.1. Experimental Validation of the Intelligent Parallel Processing. In these experiments, the validation was performed with large scales (1K-by-1K pixel format) RS images borrowed from real-world high-resolution terrain SAR imagery (lower Manhattan region, NY, USA and the Pentagon region near

Table 2: IOSNR of the aggregated RASF-PM and RASF-fuzzy anisotropic diffusion algorithms evaluated for different SNRs.

SNR (dB)	RASF-PM method (dB)	RASF-fuzzy method (dB)
5	7.13	8.15
10	7.92	10.37
15	9.75	11.74
20	10.86	13.65

Washington, DC, USA [25]). In the image formation stage, we considered the conventional side-looking synthetic aperture radar (SAR) with the fractionally synthesized aperture as an RS imaging system [11, 12]. The regular SFO **F** of such SAR is factored along two axes in the image plane using the MSF method: the azimuth or cross-range coordinate (horizontal axis, x) and the slant range (vertical axis, y), respectively. We considered the conventional triangular SAR range ambiguity function (AF) [11] and Gaussian approximation [22] of the SAR azimuth AF. Also, the chi-squared additive noise of 20 dB signal-to-noise ratio (SNR) was incorporated to test the performances of the particular RS processing chain.

In the reconstruction and post-processing stage, we conduct the evaluation of the GPU-based implementation of the well-known reconstructive and enhancement post-processing algorithms. The validation has been realized between the original scene frame, the degraded RS image, the reconstructive RASF algorithm, the Perona-Malik (PM) technique, and the fuzzy anisotropic-diffusion technique.

In analogy to the image reconstruction for quantitative evaluation of the RS reconstruction performances, the quality metric defined as an improvement in the output signal-to-noise ratio (IOSNR) is employed. This metric is defined as follows: $\text{IOSNR} = 10\log_{10}(\sum_{l=1}^{L}(\widehat{b}_l^{\text{MSF}} - b_l)^2 / \sum_{l=1}^{L}(\widehat{b}_l^{(q)} - b_l)^2)$; $q = 1, 2$, where L is related to the size of the RS image, b_l represents the value of the lth element (pixel) of the original image, $\widehat{b}_l^{\text{MSF}}$ represents the value of the lth element (pixel) of the degraded image formed applying the MSF technique, and $\widehat{b}_l^{(q)}$ represents a value of the lth pixel of the image reconstructed with two considered reconstructive-related methods; $q = 1, 2$, where $q = 1$ corresponds to the RASF-PM algorithm and $q = 2$ corresponds to the RASF-fuzzy anisotropic diffusion algorithm, respectively. The quantitative measures of the image enhancement/reconstruction performance gains achieved with the particular employed RASF-PM and RASF-Fuzzy anisotropic diffusion techniques, are evaluated via the IOSNR metric, as reported in Table 2.

According to this quality metric, the higher the IOSNR (the average over 100 realizations), the better the improvement of the image reconstruction/post-processing with the particular employed algorithm.

Next, the qualitative results are presented in Figures 4 and 5. Figures 4(a) and 5(a) show the original test scene image. Figures 4(b) and 5(b) present the noised low-resolution (degraded) scene image formed with the conventional MSF algorithm. Figures 4(c) and 5(c) present the scene image reconstructed with the RASF algorithm. Figures 4(d) and 5(d) present the scene image enhancement employing the RASF-PM post-processing algorithm, Figures 4(e) and 5(e) present the result of the fuzzy edge detection, and Figures 4(f) and 5(f) present the results of the image enhancement using the implemented RASF-fuzzy anisotropic diffusion post-processing algorithm.

From the analysis of the qualitative and quantitative simulation results reported in Figures 4 and 5 and Table 2, one may deduce that the aggregation of the reconstructive model-based RASF method and the model-free fuzzy anisotropic diffusion method overperforms the classical PM anisotropic diffusion method in the experimental validation scenario of the intelligent parallel processing toolbox.

Table 3: Comparative feature analysis of the employed GPU.

GPU features	GTS 450	Tesla C2075
Peak single precision floating point performance	601 Gflops	1030 Gflops
Memory bandwidth (ECC off)	57.7 GB/sec	148 GB/sec
Memory size (GDDR5)	1 GB	6 GB
CUDA compute capability	2.1	2.0
CUDA cores	192	448

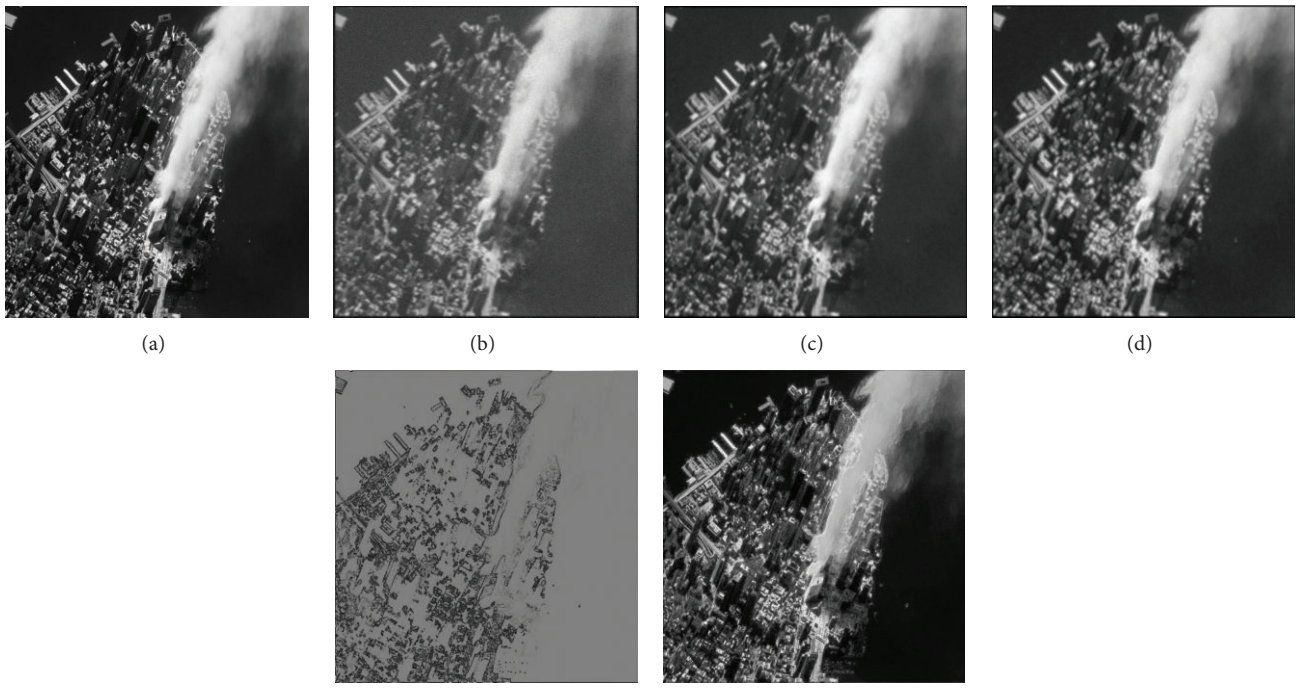
4.2. Time Performance Analysis. Next, we compared the required processing time for two different implementation schemes as reported in Tables 3 and 4. In the first case, the reference image formation/reconstruction/post-processing procedures for the intelligent parallel processing of remotely sensed imagery were implemented in the conventional C language in a Dell PowerEdge Server with an Intel Xeon E5603 Quad Core processor at 1.6 GHz and 24 GB of RAM. In the second case, the same algorithms were implemented using the proposed parallel toolbox approach which aggregates the graphic user interface of MATLAB with the parallel multiprocessor scheme based on GPU. In Table 3, a comparative feature analysis of the employed GPU is presented.

Next, considering the implementation on a Dell PowerEdge Server, which has a Nvidia Tesla C2075 GPU, we have achieved kernel-processing times up to 30 ms, which means 10+ times faster than desktop GPU processing with a Nvidia Gforce GTS 450.

The overall process for each stage of aggregation of the reconstructive model-based RASF method and the model-free fuzzy anisotropic diffusion method is 98 ms for Tesla and 897 ms for GTS, and considering only the fuzzy anisotropic kernel the times are 39 ms for Tesla and 357 ms for GTS. The speedup of each stage is over 200 times in comparison to a Matlab implementation, and 10 times a Desktop GPU implementation, as presented in the comparative processing time analysis of Table 4. Note that Matlab implementation does not need any transfers or GPU management; hence, it is only presented the overall processing time.

5. Conclusions

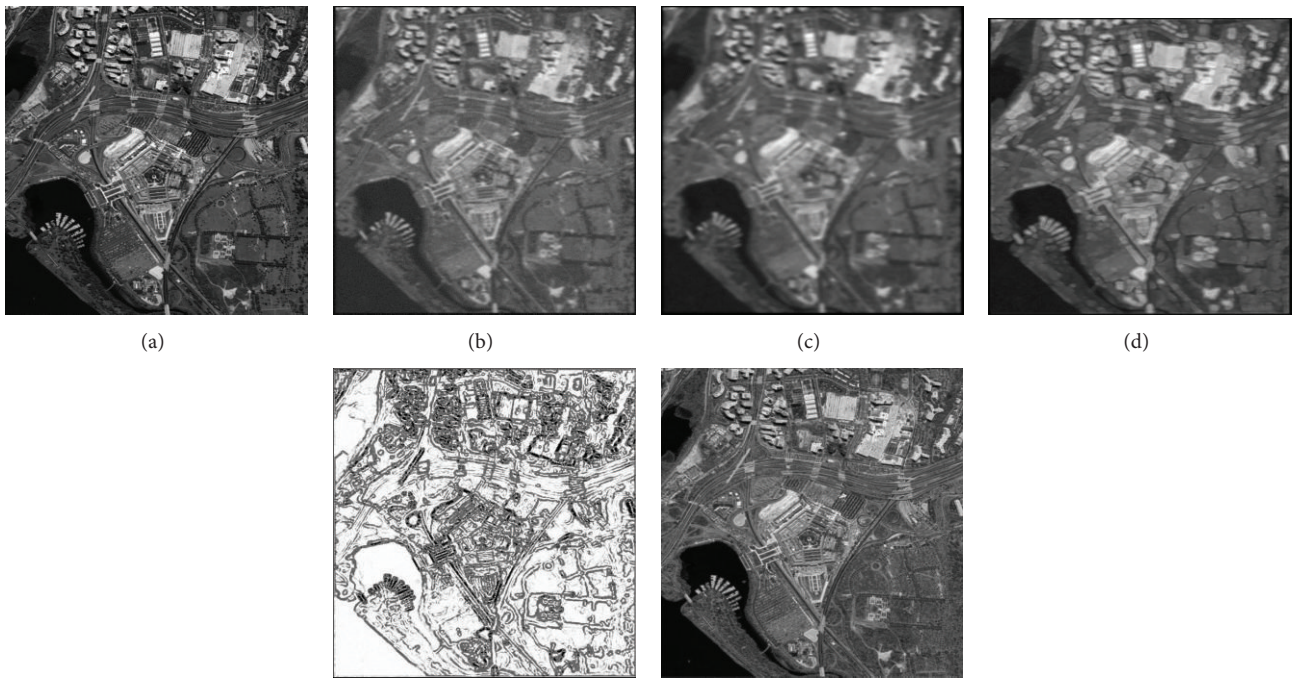
In this paper, a parallel tool for large-scale remote sensing images was presented. This tool allows the image formation



(a) (b) (c) (d)

(e) (f)

Figure 4: Experimental results: (SNR = 20 dB): (a) original test scene of lower Manhattan; (b) degraded scene image formed applying the MSF method; (c) image reconstructed applying the regularized RASF algorithm; (d) image enhancement applying the RASF-PM post-processing algorithm; (e) image applying the fuzzy edge detection; (f) image enhancement using the RASF-fuzzy post-processing algorithm.



(a) (b) (c) (d)

(e) (f)

Figure 5: Experimental results: (SNR = 20 dB): (a) original test scene which corresponds to the Pentagon region; (b) degraded scene image formed applying the MSF method; (c) image reconstructed applying the regularized RASF algorithm; (d) image enhancement applying the RASF-PM post-processing algorithm; (e) image applying the fuzzy edge detection; (f) image enhancement using the RASF-fuzzy post-processing algorithm.

Table 4: Comparative analysis of processing times for model-free fuzzy anisotropic diffusion method.

Image size	MATLAB	GTS 450 (Desktop GPU)		Tesla C2075 (HPC server)	
	Total	Total	Kernel	Total	Kernel
1 K × 1 K	9626 ms	897 ms	357 ms	98 ms	39 ms
3 K × 3 K	79670 ms	4814 ms	1916 ms	503 ms	200 ms

and the reconstruction of images using two well-known techniques: the robust spatial filter (RSF) and the robust adaptive spatial filter (RASF). It also applies a post-processing stage based on the Perona-Malik algorithm jointly with a Fuzzy Logic System (FLS) named in this paper as fuzzy anisotropic diffusion technique. This toolbox was implemented using a Matlab graphic user interface (GUI) and the mathematical operations were computed using graphics processor units (GPU). Under this paradigm, it was possible to save significant processing time due to the adaptation of the algorithms for its parallel implementation. For the presented case of studies, it was shown that the processing time in comparison with a PC based implementation was reduced +200 times. Thus, this approach represents a powerful parallel tool for processing huge amount of large scale RS images.

Acknowledgment

This study was supported by Consejo Nacional de Ciencia y Tecnologia (CONACYT), Mexico, under Grant CB-2010- 01-158136.

References

[1] "AVIRIS—Airborne Visible/Infrared Imaging Spectrometer," 2013, http://aviris.jpl.nasa.gov/index.html.

[2] R. Farber, *CUDA Application Design and Development*, Morgan Kaufmann, Waltham, Mass, USA, 2012.

[3] S. Bernabé, A. Plaza, P. Reddy Marpu, and J. Atli Benediktsson, "A new parallel tool for classification of remotely sensed imagery," *Computers and Geosciences*, vol. 46, pp. 208–218, 2012.

[4] Y. V. Shkvarko, J. Gutierrez, and L. G. Guerrero, "Towards the virtual remote sensing laboratory: simulation software for intelligent post-processing of large scale remote sensing imagery," in *Proceedings of IEEE International Geoscience and Remote Sensing Symposium (IGARSS '07)*, pp. 1561–1564, June 2007.

[5] R. H. Boucher, T. E. Dutton, S. A. Cota et al., "PICASSO: an end-to-end image simulation tool for space and airborne imaging systems," *Journal of Applied Remote Sensing*, vol. 4, no. 1, Article ID 043535, 2010.

[6] A. Castillo Atoche, D. Torres Roman, and Y. Shkvarko, "Experiment design regularization-based hardware/software codesign for real-time enhanced imaging in uncertain remote sensing environment," *EURASIP Journal on Advances in Signal Processing*, vol. 2010, Article ID 254040, 21 pages, 2010.

[7] P. Perona and J. Malik, "Scale-space and edge detection using anisotropic diffusion," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, no. 7, pp. 629–639, 1990.

[8] A. C. Atoche, Y. Shkvarko, D. T. Roman, and H. P. Meana, "Convex regularization-based hardware/software co-design for real-time enhancement of remote sensing imagery," *Journal of Real-Time Image Processing*, vol. 4, no. 3, pp. 261–272, 2009.

[9] Y. V. Shkvarko, "Unifying regularization and Bayesian estimation methods for enhanced imaging with remotely sensed data—part I: theory," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 42, no. 5, pp. 923–931, 2004.

[10] Y. Shkvarko, S. Santos, and J. Tuxpan, "Intelligent experiment design-based virtual remote sensing laboratory," in *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications*, E. Bayro-Corrochano and J. -O. Eklundh, Eds., vol. 5856 of *Lecture Notes in Computer Science*, pp. 1021–1028, Springer, Berlin, Germany, 2009.

[11] D. R. Wehner, *High-Resolution Radar*, Artech House, Boston, Mass, USA, 2nd edition, 1995.

[12] F. M. Henderson and A. J. Lewis, *Remote Sensing, Principles and Applications of Imaging Radar*, John Wiley & Sons, New York, NY, USA, 1998.

[13] Y. V. Shkvarko, "Unifying experiment design and convex regularization techniques for enhanced imaging with uncertain remote sensing data—part I: theory," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 48, no. 1, pp. 82–95, 2010.

[14] S. S. Haykin and A. Steinhardt, *Adaptive Radar Detection and Estimation*, Wiley, New York, NY, USA, 1992.

[15] T. Yardibi, J. Li, P. Stoica, M. Xue, and A. B. Baggeroer, "Source localization and sensing: a nonparametric iterative adaptive approach based on weighted least squares," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 46, no. 1, pp. 425–443, 2010.

[16] Y. V. Shkvarko, "Unifying experiment design and convex regularization techniques for enhanced imaging with uncertain remote sensing data—part II: adaptive implementation and performance issues," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 48, no. 1, pp. 96–111, 2010.

[17] Y. Gu, Y. Zhang, and J. Zhang, "Integration of spatial—spectral information for resolution enhancement in hyperspectral images," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 46, no. 5, pp. 1347–1358, 2008.

[18] A. De Maio, A. Farina, and G. Foglia, "Knowledge-aided bayesian radar detectors & their application to live data," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 46, no. 1, pp. 170–183, 2010.

[19] A. Plaza and J. Plaza, "Parallel morphological classification of hyperspectral imagery using extended opening and closing by reconstruction operations," in *Proceedings of IEEE International Geoscience and Remote Sensing Symposium*, pp. I58–I61, July 2008.

[20] J. Song and H. R. Tizhoosh, "Fuzzy anisotropic diffusion: a rule-based approach," *Proceedings of Science*, vol. 4, pp. 241–246, 2003.

[21] J. Sanders and E. Kandrot, *CUDA by Example: An Introduction to General-Purpose GPU Programming*, Addison-Wesley, Upper Saddle River, NJ, USA, 2011.

[22] H. H. Barrett and K. J. Myers, *Foundations of Image Science*, Wiley-Interscience, Hoboken, NJ, USA, 2004.

[23] D. Kirk, *Programming Massively Parallel Processors: A Hands-on Approach*, Elsevier, Amsterdam, The Netherlands, 2013.

[24] Nvidia Corporation, "NVIDIA Performance Primitives," 2013, https://developer.nvidia.com/npp.

[25] "Space Imaging, High Resolution Imagery, Earth Imagery & Geospatial Services," 2013, http://www.geoeye.com/gallery#!.