Enhancement and Edge-Preserving Denoising: An OpenCL-Based Approach for Remote Sensing Imagery

Jaime Ortegón Aguilar, *Senior Member, IEEE*, Alejandro Castillo Atoche, *Member, IEEE*, Roberto Carrasco Alvarez, Javier Vázquez Castillo, *Member, IEEE*, Ivan Villalón–Turrubiates, *Senior Member, IEEE*, and Omar Pérez-Martínez, *Student Member, IEEE*

***Abstract*—Image enhancement and edge-preserving denoising are relevant steps before classification or other postprocessing techniques for remote sensing images. However, multisensor array systems are able to simultaneously capture several low-resolution images from the same area on different wavelengths, forming a high spatial/spectral resolution image and raising a series of new challenges. In this paper, an open computing language based parallel implementation approach is presented for near real-time enhancement based on Bayesian maximum entropy (BME), as well as an edge-preserving denoising algorithm for remote sensing imagery, which uses the local linear Stein's unbiased risk estimate (LLSURE). BME was selected for its results on synthetic aperture radar image enhancement, whereas LLSURE has shown better noise removal properties than other commonly used methods. Within this context, image processing methods are algorithmically adapted via parallel computing techniques and efficiently implemented using CPUs and commodity graphics processing units (GPUs). Experimental results demonstrate the reduction of computational load of real-world image processing for near real-time GPU adapted implementation.**

***Index Terms*—Image enhancement, image processing, parallel processing, remote sensing, unmanned aerial vehicles.**

I. Introduction

SINCE the advent of remote sensing (RS) technology, there have been great changes in earth observation theory and technology. One of these is particularly related to the extraction of physical characteristics of a geographical region, such as water, land cover, vegetation, soil, and others. In this regard, the understanding of geospatial RS imagery, as well as the solution of diverse pattern recognition tasks, drastically depends on the provided RS image quality, acquired from multisensor array systems, such as radar/synthetic aperture radar (SAR) or unmanned aerial vehicles (UAVs) [1]–[3]. The enhancement of noise-corrupted RS imagery before classification analysis is a crucial information processing task that could substantially improve decision making results.

In this study, the enhancement approach is stated and treated as an uncertain ill-posed inverse problem of nonparametric estimation of the power spatial spectrum pattern (SSP) of the wavefield scattered from a remotely sensed scene. These operational uncertainties are associated with the unknown statistics of perturbations of the adjoint signal formation operator (SFO) of the recorded data in the turbulent medium, finite dimensionality of measurements, uncontrolled UAV vibrations, and random carrier trajectory deviations in the case of SAR. Moreover, speckle noise and possible calibration errors constitute additional multiplicative sources of data degradations that inevitably aggravate the inconsistency problem resulting in heavily distorted speckle-corrupted scene images. To derive the estimate of the SSP, the Bayesian maximum entropy (BME) strategy is applied for maximization of the *a posteriori* probability density function (pdf) of the randomized maximum entropy (ME) model of the SSP[4]. The BME is a statistical optimization method algorithmically adapted for RS systems with robust *a priori* information about the statistics of noise. The development of this method employs the ME robust regularization of the nonlinear multispectral imagery. The estimator is a nonlinear adaptive algorithm that permits a concise and robust implementation. However, high-frequency image components and the current image enhancement/reconstruction techniques oriented for large-scale RS imaging do not work well for edge-preserving smoothing of images corrupted with additive and speckle noise. To reduce these effects, a postprocessing framework is proposed with the local linear Stein's unbiased risk estimate (LLSURE) based edge-preserving image filtering technique, which preserves image details and local geometries while removing undesirable noise [5]. The LLSURE is a filter based on Stein's unbiased risk estimate (SURE); Jain and Tyagi [6] carried out several test confirming it to be among the best edge-preserving methods for removing noise.

Additionally, another scientific challenge of this study consists of how to implement the enhancement and edge-preserving denoising tasks on large-scale multispectral images. In this sense, recent research efforts have been directed toward the incorporation of parallel computing techniques and the design of specialized high-performance computing (HPC) architectures for intelligent processing of remotely sensed images with a high spatial–spectral resolution. A good review on HPC applied to RS is presented in [7], where its potential and challenges are analyzed. Other implementations of RS image processing using low-cost graphics processing units (GPUs) are presented in [8]–[10]. A framework for GPU adaptation of image processing algorithms is presented in [11], where Christophe *et al.* consider issues arising from hardware restrictions. On the other hand, the open computing language (OpenCL) is used for tackling highly demanding data processing, either with a parallel data processing approach, such as satellite image segmentation [12], spatial data interpolation [13], vertex component analysis [14], or parallel task scheduling approach[15]. However, theoretical and practical data-processing challenges to developing HPC solutions based on massively parallel hardware accelerators remain unsolved in many applications. Therefore, a novel OpenCL-based approach for near real-time enhancement and edge-preserving denoising has been developed in this study. OpenCL provides an industry standard for parallel programming of heterogeneous computing platforms, and is designed to exploit the compute capability of devices, such as CPUs (central processing units), FPGAs (field programmable gate arrays), GPUs, ARM SoC [advanced RISC (reduced instruction set computer) machines system on a chip] with general-purpose GPUs, and others. The approach introduced in this paper is based on the use of image enhancement and edge-preserving denoising techniques with the following design flow: 1) image reconstruction via BME algorithm, and 2) postprocessing edge-preserving denoising filtering through the principle of LLSURE. Moreover, this approach is also related to the efficient implementation of the presented framework using multi-GPU devices with the aim of accelerating high demand operations embedded in the analysis of a particular geographical region. Thus, although different high-performance computing systems have been proposed for RS multispectral imagery via algorithm implementation in commodity clusters, FPGA-based systems, or digital signal processor systems; the authors consider that the presented approach based on the unification of the aforementioned techniques and the use of the parallel OpenCL-based standard will provide high accuracy in the enhancement and edge-preserving denoising process, allowing a near real-time data processing.

The rest of the paper is organized as follows. In Section II, the problem statement and the algorithm description are presented. The GPU implementation of the multispectral image system is described in Section III. Next, the implementation and performance analysis are discussed in Section IV; one synthetic and two real-world case study scenarios are presented to show the reduction in the processing time of the proposed enhancement and edge-preserving denoising algorithm. Finally, the concluding remarks are presented in Section V.

II. RS Imaging Problem Model

In this section, a brief description of the general formalism of the RS problem and the proposed design flow is presented.

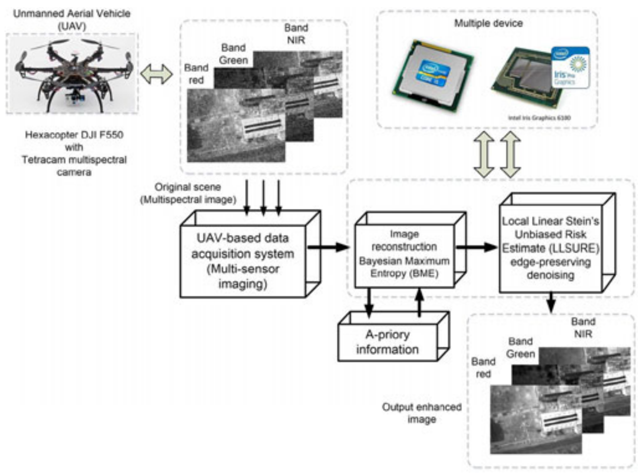
Unmanned Aerial Vehicle (UAV)
Hexacopter DJI F550 with Tetracam multispectral camera
Band red
Band Green
Band NIR
Multiple device
Intel Iris Graphics 6100
Original scene (Multispectral image)
UAV-based data acquisition system (Multi-sensor imaging)
Image reconstruction Bayesian Maximum Entropy (BME)
Local Linear Stein's Unbiased Risk Estimate (LLSURE) edge-preserving denoising
A-priory information
Band red
Band Green
Band NIR
Output enhanced image

Fig. 1. Multispectral RS imaging problem model.

A complete study related to this relevant topic can be found in [16] and [17].

A. Problem Formulation

This subsection is focused on the modeling of the multispectral RS imaging problem. Fig. 1 illustrates the RS problem model particularly oriented to the extraction, reconstruction, and postprocessing analysis of environmental features for applications in natural resources management.

Considering the analysis of Fig. 1, the results are adopted from previously developed data acquisition and image enhancement techniques [4], [18]–[20], for high-resolution reconstruction of the power SSP of the wavefield scattered from the remotely sensed scene (that is referred to as the desired RS image $\hat{\mathbf{B}}$ [4]) given a finite set of multispectral UAV signal recordings.

In this particular study, the challenge is to implement the unified enhancement and edge-preserving denoising algorithms from the UAV-based multispectral scene using GPU computing. The proposed OpenCL parallel design enables us to enhance the corrupted scene with additive and speckle noise in near real time.

B. Theoretical Background

Following the multispectral RS imaging problem of Fig. 1, the data processing blocks (referred to *RS acquisition system, image reconstruction, and edge-preserving denoising*) are briefly described.

1) RS Data Acquisition System: Multispectral sensors acquire image data, allowing physical measurements. These sensors collect image data simultaneously in the visible and near-infrared (NIR) radiation reflected from the earth's surface. In this paper, the multispectral images are acquired with an ADC-lite (analog-to-digital converter) Tetracam camera integrated in a UAV system with high spatial resolution images [3.2 megapixel CMOS (complementary metal-oxide-semiconductor) sensor] with green, red, and NIR bands comparable to Landsat bands 2, 3, and 4 [21]. This particular image has a spatial resolution of 685 mm^2 per pixel when acquired at an altitude of 83 m.

2) Image Reconstruction: Once the multispectral image is acquired, the measurement data wavefield $u(y) = s(y) + n(y)$ of each band is modeled as a superposition of the echo signals s and additive noise n, assumed to be available for observations and recordings within the prescribed time-space observation domain $Y \ni \mathbf{y}$, where $\mathbf{y} = (t, \mathbf{y})^{\mathrm{T}}$ defines the time-space points in the observation domain. The conventional finite-dimensional vector-form approximation of the RS data observation is given by

$$\mathbf{u} = \mathbf{Se} + \mathbf{n} \tag{1}$$

where $\mathbf{u}, \mathbf{n}$, and $\mathbf{e}$ are composed vectors of the finite dimensional approximations of the measurement field u, the observation noise n, and the scene scattered field e, respectively, and $\mathbf{S}$ is the matrix-form approximation of the SFO S, specified by the particular modulation format employed in the RS system in [18] and [4].

The RS imaging problem is now stated as follows: to find an estimate of each band of the scene pixel frame $\hat{\mathbf{B}}$ via lexicographical reordering $\hat{\mathbf{B}} = L\{\hat{\mathbf{b}}\}$ of the estimated SSP vector $\hat{\mathbf{b}}$, which is reconstructed from the available measurements of independent realizations $\{\mathbf{u}_{(j)}; j = 1, \ldots, J\}$ of the recorded data vector. The lexicographical reordering $L\{\cdot\}$ is a reshaping of a vector into a matrix. Here, $\mathbf{b}$ is referred to the SSP, which represents the brightness reflectivity of the RS scene [18]. This image reconstruction is achieved using the BME algorithm. The following assumptions are considered: vector $\mathbf{b}$ is viewed as an element of the K-D vector space $\mathcal{B}_{(K)} \ni \mathbf{B}$ with the squared norm imposed by the inner product $\|\mathbf{B}\|^2_{(K)} = [\mathbf{B}, \mathbf{MB}]$, where $\mathbf{M}$ is a positive definite weighting matrix. Usually, $\mathbf{M}$ is selected as the matrix-form approximation of some differential operator, in which case, prior information about the smoothness properties of the desired SSP is embedded in $\mathbf{M}$. The metrics structure in $\mathcal{B}_{(K)}$ is determined by this weighting matrix $\mathbf{M}$; hence, its selection provides the additional degrees of freedom of the problem model. According to the ME principle of information theory [3], the *a priori* pdf $p(\mathbf{B})$ is employed, inserting the minimum amount of information about the SSP, but taking into account the nontrivial model knowledge, formalized as follows:

$$\int_{B_C} [\mathbf{B}, \mathbf{MB}] p(\mathbf{B}) d\mathbf{B} \leq c_0 \tag{2}$$

where the normalized constant c_0 is usually unknown.

Following the ME model, $p(\mathbf{B})$ is found, maximizing the entropy integral $-\int \ln p(\mathbf{B})\, p(\mathbf{B}) d\mathbf{B}$, where the maximization must be found as a solution to the Lagrange maximization problem as follows:

$$\begin{aligned} \max_{p(\mathbf{B})} \rightarrow \; & \int_{B_C} \ln p(\mathbf{B})\, p(\mathbf{B}) d\mathbf{B} \\ & - \alpha \left(\int_{B_C} [\mathbf{B}, \mathbf{MB}] p(\mathbf{B}) d\mathbf{B} - c_0 \right) \\ & - \lambda \left(\int_{B_C} p(\mathbf{B}) d\mathbf{B} - 1 \right) \end{aligned} \tag{3}$$

with the Lagrange multipliers α, λ. Performing the maximization described in (3) and following the Gibbs *a priori* pdf, the solution to the Lagrange problem is

$$p(\mathbf{B}|\alpha) = \exp\Big(-\ln \sum(\alpha) - \alpha[\mathbf{B}, \mathbf{MB}]\Big), \qquad \mathbf{B} \in B_C \tag{4}$$

with the scalar α dependent on the constant c_0 and $\sum(\alpha)$ is the Boltzmann statistical sum [3].

Now, let us define the log likelihood function $\Lambda(\mathbf{B}|\mathbf{U})$ of vector $\mathbf{B}$ oriented to solve the BME, estimating the SSP as follows:

$$\hat{\mathbf{B}} = \arg\min_{\mathbf{B},\alpha} \{-\Lambda(\mathbf{B}|\mathbf{U}) - \ln p(\mathbf{B}|\alpha)\} \tag{5}$$

where the Gibbs prior distribution $p(\mathbf{B}|\alpha)$ is defined by (4). Applying the standard gradient method to minimize (5), the desired BME of the SSP can be found as a solution of the nonlinear equation

$$\hat{\mathbf{B}} = \mathbf{W}(\hat{\mathbf{B}})[\mathbf{V}(\hat{\mathbf{B}}) - \mathbf{Z}(\hat{\mathbf{B}})] \tag{6}$$

where $\mathbf{V}(\hat{\mathbf{B}})$ is a vector that represents the sufficient statistics (SS) for the Bayesian estimator of the SSP; vector $\mathbf{Z}(\hat{\mathbf{B}})$ has the statistical meaning of a shift or bias vector; and $\mathbf{W}(\hat{\mathbf{B}})$ has the statistical meaning of a solution-dependent (i.e., adaptive) window operator. Note that the derivation of this SSP estimator is detailed in [3], and references therein.

Nonlinearity of (6) also implies the dependence of $\mathbf{V}, \mathbf{Z}$, and $\mathbf{W}$ on the desired estimate $\hat{\mathbf{B}}$, and thus, no unique regular method for solving (6) exists. This means that implementation of the optimal estimator inevitably requires solution-dependent processing, which is referred to as *adaptive* with respect to the desired estimate. Nevertheless, it is possible to perform the robustification (nonadaptive approximation) of the BME algorithm. Following the robust regularization idea of [22], the algorithm is adapted, reducing the computational load of some relevant expressions of the BME estimator. In this sense, the robust SS vector is now computed as

$$\mathbf{V} = \{\mathbf{FUU}^{+}\mathbf{F}^{+}\}_{\text{diag}} \tag{7}$$

where $\mathbf{F} = (\mathbf{S}^{+}\mathbf{S} + \varpi^{-1}\mathbf{I})\mathbf{S}^{+}$, with regularization parameter $\varpi = \frac{\beta}{N_o}$, which represents the signal-to-noise ratio. The robust smoothing window

$$\mathbf{W} = (w_0\, \mathbf{I} + \mathbf{M})^{-1} \tag{8}$$

is completely defined by the matrix $\mathbf{M}$, which induces the metrics structure in the solution space with the scaling factor $w_0 = \text{tr}\{\mathbf{S}_{+}\mathbf{F}_{+}\mathbf{FS}\}/K$, and the bias vector $\mathbf{W} = \{\mathbf{FR}_{\text{N}}\mathbf{F}_{+}\}_{\text{diag}}$ can be neglected, as it does not affect the pattern of the SSP estimate (it influences only the constant gray level in the resulting solution). Here, $\mathbf{R}_{\text{N}} = N_0\mathbf{I}$ represents the adopted observation white noise model. Thus, in the framework of the Bayesian inference based approach to estimating the wavefield power distribution in the remotely sensed environment, the estimator

$$\hat{\mathbf{B}} = \mathbf{WV} \tag{9}$$

may be viewed as a rough simplified version of the BME method developed above. However, the computational complexity of the BME rough simplified version still represents a challenge in this study.

C. Local Linear Stein's Unbiased Risk Estimate (LLSURE)

The LLSURE [5] is a filter based on SURE. To understand how it works, we first need to know Stein's lemma, which could be classified in two variants: the univariate and the multivariate lemma [23]. We will explain the multivariate variant, considering $n = 2$, since images are two-dimensional (2-D) signals.

Lemma Let $X \sim N(\mu, \sigma^2 I)$, a 2-D normal variate, with a mean of $\mu \in \mathbb{R}^2$ and a spherical covariance matrix of $\sigma^2 I \in \mathbb{R}^{2\times 2}$, and let $f : \mathbb{R}^2 \to \mathbb{R}$ be a function such that for each $i = 1, 2$, and almost every $x_{-i} \in \mathbb{R}$, the function

$$f(\cdot, x_{-1}) : \mathbb{R} \to \mathbb{R} \tag{10}$$

is absolutely continuous and is known as almost differentiable. It is important to note that this function has partial derivatives almost everywhere, and their collection is denoted with $\nabla f = (\vartheta f/\vartheta x_1, \ldots, \vartheta f/\vartheta x_n)$. Stein's result with such X, and almost differentiable f, is

$$\frac{1}{\sigma^2}\mathbb{E}\left[(X-\mu) f(X)\right] = \mathbb{E}\left[\nabla f(X)\right]. \tag{11}$$

If $f = (f_1, \ldots, f_n)$, i.e., the coordinate functions, then for each $i = 1, \ldots, n$

$$\frac{1}{\sigma^2}\mathbb{E}\left[(X-\mu) f_i(X)\right] = \mathbb{E}\left[\nabla f_i(X)\right]. \tag{12}$$

Summing $i = 1, \ldots, n$ gives

$$\begin{aligned}\frac{1}{\sigma^2}\sum_{i=1}^{n}\text{Cov}\left(X_1, f_1(X)\right) &= \frac{1}{\sigma^2}\sum_{i=1}^{n}\mathbb{E}\left[(X_i-\mu_i) f_1(X)\right] \\ &= \mathbb{E}\left[\sum_{i=1}^{n}\frac{\partial f_1}{\partial x_1}(X)\right].\end{aligned} \tag{13}$$

Now, considering $\hat{\mu}$ as an estimate of mean μ, the risk is computed as

$$R = \mathbb{E}\left\|\mu - \hat{\mu}\right\|^2 = -n\sigma^2 + \mathbb{E}\left\|y-\hat{\mu}\right\|^2 + 2\sigma^2 \text{dof}(\hat{\mu}) \tag{14}$$

where $\mathbb{E}\left\|y-\hat{\mu}\right\|^2$ is the expected training error of $\hat{\mu}$ and its degrees of freedom are defined as

$$\text{dof}(\hat{\mu}) = \frac{1}{\sigma^2}\sum_{i=1}^{n}\text{Cov}(y_i, \hat{\mu}_i). \tag{15}$$

Stein's lemma provides an explicit estimation of the degrees of freedom term $\text{dof}(\hat{\mu})$ and also the risk of R. Knowing that $\hat{\mu}$ is almost differentiable as a function of f, we have

$$\hat{R} = -n\sigma^2 + \left\|y-\hat{\mu}\right\|^2 + 2\sigma^2\sum_{i=1}^{n}\frac{\partial\hat{\mu}_i}{\partial y_i}(y). \tag{16}$$

This expression is known as SURE. In the case of the estimator depending on a tuning parameter $\lambda \epsilon \Lambda$, it is denoted $\hat{\mu}_\lambda$ and it is possible to choose this parameter to minimize SURE

$$\hat{\lambda} = \underset{\lambda\in\Lambda}{\text{argmin}}\left\|y-\hat{\mu}_\lambda\right\|^2 + 2\sigma^2\sum_{i=1}^{n}\frac{\partial\hat{\mu}_{\lambda,i}}{\partial y_i}(y). \tag{17}$$

The mean-squared error of the denoised image with respect to its noise-free version is

$$\text{MSE}(\hat{x}) = \frac{1}{N}\left\|x-\hat{x}\right\|^2 = \frac{1}{N}\sum_{i=1}^{N}(x_i - \hat{x}_i)^2 \tag{18}$$

where $\|\cdot\|^2$ is the Euclidean norm. Now, applying the SURE, we have

$$\text{SURE}(\hat{x}) = \frac{1}{N}\left(\left\|y-\hat{x}\right\|^2 + 2\sigma^2\sum_{i=1}^{N}\frac{\partial\hat{x}_i}{\partial y_i}(\hat{x}) - N\sigma^2\right). \tag{19}$$

In a local linear model, it is assumed that a filtered output image patch is an affine transformation of an input image patch in the same position. The latter is true for a local neighborhood around every pixel. It is possible to determine the optimal transform coefficients by minimizing the MSE estimate with SURE. Therefore, let ω_i be a local window around the ith position, and y_ω and $\hat{x}_\omega$ be an input and filtered output image patch, respectively, corresponding to window ω_i, then

$$\hat{x}_{\omega_i} = a_i y_{\omega_i} + b_i, \quad a_i, b_i \in \mathbb{R}, \quad a_i \geq 0 \tag{20}$$

where a_i and b_i are the affine coefficients assumed to be constants in window ω_i, thus

$$\forall j \in \omega_i, \hat{x}_j = a_j y_j + b_j. \tag{21}$$

Using (20) in (19), we have

$$\begin{aligned}\text{SURE}(a_i, b_i) &= \frac{1}{N_w}\left\|y_{\omega_i} - (a_i y_{\omega_i} + b_i)\right\|^2 + \frac{2\sigma^2}{N_\omega}a_i N_\omega - \sigma^2 \\ &= \frac{1}{N_\omega}\left\|y_{\omega_i} - (a_i y_{\omega_i} + b_i)\right\|^2 + 2\sigma^2 a_i - \sigma^2\end{aligned} \tag{22}$$

where $N_\omega = (2r+1)(2r+1)$ is the pixel number in window ω_i, and r is the window radius.

To compute the optimal affine transform coefficients a_i and b_i of local window ω_i, it is necessary to take the first derivatives of (22) with respect to a_i and b_i and to consider that a_i is restricted as nonnegative. The resulting optimal affine coefficients are

$$a_i = \max(\sigma_i^2 - \sigma^2, 0)/\left(\sigma_i^2 + \varepsilon\right) \tag{23}$$

$$b_i = (1 - a_i)\,\overline{y_i} \tag{24}$$

where $\overline{y_i}$ and σ_i^2 are the mean and variance, respectively, of the input data in local window ω_i, and ε is a small constant to avoid division by zero.

Moving local window ω_i over the entire image, it is possible to compute the corresponding affine coefficients. However, each pixel may be included in several windows, due to the overlap of each moved window. Hence, considering that pixels x_j in overlapping regions have multiple estimates $\hat{x}_j^i$, it is necessary to add them into the final estimate

$$\hat{x}_j = \sum_{i\in w_j}\lambda_i \hat{x}_j^i \tag{25}$$

where λ_i is the weight for the ith estimate. As in [5], to compute λ_i, we use the following equation, inspired by the variance-based

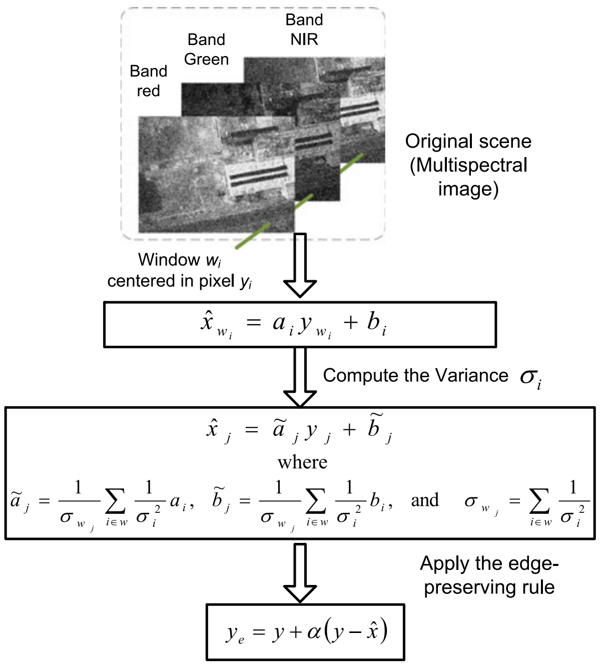
Band NIR
Band Green
Band red
Original scene (Multispectral image)
Window w_i centered in pixel y_i
$\hat{x}_{w_i} = a_i y_{w_i} + b_i$
Compute the Variance σ_i
$\hat{x}_j = \tilde{a}_j y_j + \tilde{b}_j$
where
$\tilde{a}_j = \frac{1}{\sigma_{w_j}} \sum_{i \in w} \frac{1}{\sigma_i^2} a_i$, $\tilde{b}_j = \frac{1}{\sigma_{w_j}} \sum_{i \in w} \frac{1}{\sigma_i^2} b_i$, and $\sigma_{w_j} = \sum_{i \in w} \frac{1}{\sigma_i^2}$
Apply the edge-preserving rule
$y_e = y + \alpha(y - \hat{x})$

Fig. 2. Algorithm to estimate $\hat{x}$ and the final y_e.

weighted average

$$\lambda_i = \frac{\sigma_i^{-2}}{\sum_{k \in w_j} \sigma_k^{-2}}. \tag{26}$$

The final LLSURE filter is

$$\begin{aligned}\hat{x}_j &= \sum_{i \in w_j} \lambda_i \hat{x}_j^i = \sum_{i \in w_j} \frac{\sigma_i^{-2}}{\sum_{k \in w_j} \sigma_k^{-2}} (a_i y_j + b_i) \quad (27)\\ &= \frac{1}{\sum_{k \in w_j} \sigma_k^{-2}} \sum_{i \in w_j} (\sigma_i^{-2} a_i y_j + \sigma_i^{-2} b_i)\\ \hat{x}_j &= \tilde{a}_j y_j + \tilde{b}_j \quad (28)\end{aligned}$$

where $\tilde{a}_j = \frac{1}{\sigma_{w_j}} \sum_{i \in w} \sigma_i^{-2} a_i$, $\tilde{b}_j = \frac{1}{\sigma_{w_j}} \sum_{i \in w} \sigma_i^{-2} b_i$, and $\sigma_{w_j} = \sum_{i \in w} \sigma_i^{-2}$.

The LLSURE filter can be used for several purposes, such as image denoising, joint denoising, detail smoothing and enhancement, high dynamic compression, and others. In this study, we will use it for edge preserving and enhancement. To do so, the achieved detail level is obtained as

$$\delta = y - \hat{x} \tag{29}$$

where y is the original image, y_σ is the output filtered image by LLSURE, and δ is the difference between the images. The detail enhanced image measurement is finally computed as

$$y_e = y + \alpha\delta \tag{30}$$

where α is the scale factor. The algorithm followed to compute the estimated $\hat{x}$ and the final y_e is depicted in Fig. 2.

TABLE I
OPENCL SETUPPROCEDURE EXECUTED IN THE HOST

1. Initialization
Obtain a list of available platforms and devices Create context, command queue and memory objects Read kernel file Create program object Compile kernel Create kernel object Set kernel arguments
2. Execution
Enqueue task, i.e., tell device to execute kernel instance If necessary, wait for previous events to complete
3. Finalization
Read memory object Free objects

III. EFFICIENT OPENCL-BASED IMPLEMENTATION

In this section, the methodology of the efficient BME and LLSURE algorithm's implementation are presented. Therefore, considering the proposed aggregation of parallel techniques with multi-GPU computing, we distinguish four stages in the design methodology for the objective of this paper:

1) OpenCL setup procedures;
2) OpenCL implementation of BME;
3) OpenCL implementation of LLSURE method;
4) Multidevice BME-LLSURE implementation (optional).

A. *OpenCL Setup*

OpenCL is an open standard for cross-platform, heterogeneous parallel programming [24]. It is supported in several processors, such as CPU, GPU, FPGA, and ARM. The OpenCL execution model is based on concurrent execution of kernel instances over a virtual grid defined by the host. OpenCL defines a context, i.e., a space, where devices receive kernels and transfer data; a context may include more than one device. Each device has one or more compute units composed of one or more processing elements and local memory. The virtual grid is a collection of work groups, which are collections of work items that execute on a single compute unit. The work items in the group execute the same kernel and share local memory and work-group barriers. OpenCL uses commands to execute kernels, reading and writing memory objects. A kernel is a function implementing the "parallel" code to be executed on an OpenCL device. An instance of a kernel to be executed is a task.

Setup procedures: As with other massively parallel hardware accelerators platforms, an OpenCL program requires working on both, the host side and the device side. The host program selects the device, and allocates and frees resources for it; the device executes the parallel program (kernel) using the OpenCL runtime API (application programming interface). Usually, the host is programmed in C/C++ using an OpenCL runtime API, and the device is programmed in OpenCL C. The setup procedure, which is executed in the host, includes initialization, execution, and finalization. This involves several steps, which are listed in Table I.

TABLE II
ALGORITHMIC SUMMARY OF THE BME METHOD

1. Mathematical problem model		
Equation of observation	$\mathbf{u} = \mathbf{S}\mathbf{e} + \mathbf{n}$	(1)
Maximum entropy strategy	$\hat{\mathbf{B}} = \arg\min_{\mathbf{B},\alpha}\{-\Lambda(\mathbf{B}\|\mathbf{U}) - \ln p(\mathbf{B}\|\alpha)\}$	(5)
BME adopted model	$\hat{\mathbf{B}} = \mathbf{W}(\hat{\mathbf{B}})[\mathbf{V}(\hat{\mathbf{B}}) - \mathbf{Z}(\hat{\mathbf{B}})]$	(6)
2. *A priori* information		
Metrics inducing operator	$\mathbf{M} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 0 & -1 & 1 \end{bmatrix}$	(8)
White observation noise	$\mathbf{R}_{\mathrm{N}}$	(8)
Regularization parameter	ϖ	(8)
3. Robust BME strategy		
Robust SS vector	$\mathbf{V} = \{\mathbf{F}\mathbf{U}\mathbf{U}^{+}\mathbf{F}^{+}\}_{\text{diag}}$ with $\mathbf{F} = (\mathbf{S}^{+}\mathbf{S} + \varpi^{-\mathbf{1}}\mathbf{I})\mathbf{S}^{+}$	(7)
Robust smoothing window	$\mathbf{W} = (w_0\,\mathbf{I} + \mathbf{M})^{-1}$ and $w_0 = \mathrm{tr}\{\mathbf{S}_{+}\mathbf{F}_{+}\mathbf{F}\mathbf{S}\}/K$	(8)
4. BME approach		
BME robustified model	$\hat{\mathbf{B}} = \mathbf{W}\mathbf{V}$	(9)

All steps are required, but some may be implicit, e.g., if available platforms and devices are known or there was an offline compilation. The selection of device is carried out when the context is created; it is at this point that the user decides to use one of the different types of hardware accelerators, such as single or multicore CPU, GPU, or other OpenCL accelerators.

B. *OpenCL Implementation of BME*

In this stage, the desired BME implementation of the SSP estimation is performed as a solution of the nonlinear equation presented in (6). This massively parallel approach is designed to speed up the complex reconstructive signal processing operations of the algorithm aimed to meet the near real-time imaging system requirements. From the analysis of (6) and (9), system-level partitioning functions are specified with the OpenCL-based paradigm implemented in a parallel form.

The design methodology is the following, according to Table II: In order to solve the linear inverse problem of the model given in (9), and according to the ME principle of information theory, the data observations $\mathbf{u}$, and the additive white Gaussian noise $\mathbf{n}$, a solution operator $W : U \longrightarrow V$ can be derived as an optimal estimate of the SSP. In this regard, the SFO $\mathbf{S}$ is constructed first to proceed with this regularization technique using the BME method. Next, the metric inducing matrix operator $\mathbf{M}$, the white observation noise model $\mathbf{R}_{\mathrm{N}}$, and the regularization parameter ϖ, provide additional knowledge about the problem. Now, the robust SS vector $\mathbf{V}$ and the robust smoothing window $\mathbf{W}$ with the scaling factor w_0 are implemented in parallel. In the robust smoothing window, the matrix inversion operation based on the block-Lower-Upper matrix decomposition (LU) decomposition is employed to solve the robustified BME approach.

The OpenCL implementation of the robustified BME strategy considers using both CPU and GPU architectures in a massively parallel scheme. A serial host code in the CPU runs in the host and a parallel kernel code runs in several device threads across multiple GPU cores. As can be deduced from Algorithm 1, the estimator is constructed while simultaneously leveraging the GPU hardware features. The image is divided and distributed between GPU and CPU processor cores using the partitioning criteria of Section III-D, and the threads within a block work together efficiently, exchanging data via a local-shared memory. The header of kernel is

```
__kernel void BME (__global const uchar*
restrict inputImage, const int width,
const int height, const int omega,
__global float* restrict F, __global
float* restrict S, __global float*
restrict outputImage);
```

C. *OpenCL Implementation of LLSURE Method*

In this stage, OpenCL kernels are implemented to compute the LLSURE of each pixel of the multispectral image. This processing stage is highly improved due to its potential for parallelization in a massively parallel scheme. Based on Fig. 2, we decided that LLSURE kernels would be launched once for each band, with as many threads as rows in the multispectral image, where each thread executes the corresponding operation of the LLSURE method.

It is necessary to use three kernels; the first one, "LLSURE_sigma," computes the variance for each window and an initial value of a_i and b_i to apply (22). The second one, "LLSURE_lambda," is used to compute weights λ_i and final values $\bar{a}_i$ and $\bar{b}_i$. The last is used to normalize the resulting image, so pixels are integers in the 0–255 range.

1) LLSURE_sigma: This kernel is used to compute the mean and variance of image window ω_i; this can be carried out with a convolution implementing a well-known parallelization scheme. Afterward, a_i and b_i are computed with (23) and (24), respectively. In order to preserve values for the next kernel, the memory is allocated on a device and the pointers are sent as arguments to the kernel. The header of the kernel is

```
__kernel void LLSURE_sigma(__global const
uchar* restrict inputImg, const int width,
const int height, __global float* restrict
a, __global float* restrict b, __global
float* restrict sigma);
```

It is important to mention that it is necessary that kernel LLSURE_sigma finish before starting the next kernel LLSURE_lambda, in order to use the correct σ_i, a_i, and b_i values; this is done by means of barriers or events. Barriers are OpenCL functions to synchronize work-item execution, which require that all work items in the same work group execute before continuing beyond the barrier; they are useful in the event that a single accelerator device is used. However, in the case of two or more devices, it is necessary to use events, which follow a consumer–producer pattern where each task produces an event and the consumer waits until all events are completed before starting its own execution. Both synchronization methods are used in the proposed implementation.

2) LLSURE_lambda: This kernel is used to apply the weights λ_i of (26); this is done by implementing (28). First, the sums $\sum_{i\in w}\sigma_i^{-2}a_i$, $\sum_{i\in w}\sigma_i^{-2}b_i$, and $w_j = \sum_{i\in w}\sigma_i^{-2}$ are computed using a convolution parallelization scheme. Afterward, the first two sums are divided by the last to obtain $\bar{a}$ and $\bar{b}$, respectively. Finally, the final estimation is computed using (28). Again, it is necessary to synchronize the execution of work items before continuing to the next kernel. This kernel uses the previously computed a, b, and σ, and returns a new array of values outputImage. The kernel's header is

```
__kernel void LLSURE_lambda(const int
width, const int height, __global const
float* restrict a, __global const float*
restrict b, __global const float* restrict
sigma, __global float* restrict outputIm-
age);
```

3) LLSURE_normal: This kernel is used to normalize the estimated values to an interval meaningful for unsigned char data. It is required to compute minimum and maximum values, which could be performed with a "parallel reduction" scheme [25]. Finally, a simple normalization formula, $finalImage_i = (outpuImage_i - min)/(max - min)$, is applied. Again, it is necessary to synchronize the execution of work items before continuing to the next kernel. This kernel uses the previously computed outputImage, returning the finalImage. The header of the kernel is

```
__kernel void LLSURE_normal(const int
width, const int height, __global const
float* restrict outputImage, __global
uchar* restrict finalImage);
```

4) LLSURE_edges: The LLSURE filter can be edge preserving with the use of (30). This can be done using a naive parallel scheme, sending each pixel to a work item. The header of the kernel is

```
__kernel void LLSURE_edges(const int
width, const int height, const int factor,
__global const uchar* restrict inputImage,
__global const uchar* restrict finalImage,
__global uchar* restrict edgesImage);
```

D. Multidevice BME-LLSURE Implementation

In this section, the implementation of the BME-LLSURE algorithm with multiple devices is described. Opposite to the implementation with a single device, where OpenCL kernels distribute the computational load on one accelerator (CPU or GPU), with multiple devices, it is possible to encounter three scenarios.

1) One device per context. This is the simplest way to access all devices, since the platform is independent, but the synchronization between devices requires copying data to host memory.
2) Multiple devices in one context (devices of same platform/vendor). There must be a queue for each device, but it facilitates sharing data between devices. Usually, it is difficult to guarantee that all devices are from the same vendor.

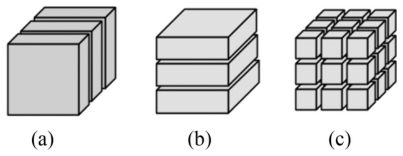

Fig. 3. Types of multispectral image data partitioning: (a) spectral, (b) spatial, and (c) mixed.

3) Multiple devices from the same platform in one context, one context per platform. This is the most flexible alternative, but also a tricky one, since the work distribution has to be well thought-out because of its two levels.

The authors chose to use only one context, taking advantage of the platform and devices used, since both may use the host unified memory, i.e., it is not necessary to explicitly transfer data to the device. In this case, a data parallel model is used dividing the data over all available devices. In order to achieve such a distribution, it is necessary to perform a partition of the multispectral image data between the devices.

E. Spatial–Spectral Partitioning

Three different types of data partitioning are identified for multispectral image processing [26]: spectral, spatial, and mixed, which are illustrated in Fig. 3. Considering the previous partitioning scheme, we select the spectral partition, as the LLSURE requires windows on the same spectral band. For large images, an overflow could occur in the device memory; to address this problem, it is necessary to query the device capabilities and divide the image accordingly. Next, Algorithm 1 is queued in every device with their corresponding portion (bands) of the multispectral image.

Let us consider a multispectral image of $x \times y$ pixels with z bands and a platform of n GPUs and m CPUs. The image ratio to be processed by one GPU (γ) is represented by

$$\gamma = \frac{1 - m\beta}{n} \tag{31}$$

where $\beta \in [0, 1]$ represents the image ratio to be processed by one CPU. The processing ratio ρ, defined by

$$\rho = \frac{\gamma}{\beta} \tag{32}$$

represents the proportion of pixels that process the GPU in regard to CPU.

Now, substituting (31) in (32) and solving for β, the image ratio to be processed by the CPU is now defined as

$$\beta = \frac{1}{n\rho + m}. \tag{33}$$

In order to compute β, it is necessary to provide a value of ρ. If it is assumed that the time processing of both CPU and GPU is directly proportional to the amount of pixels, then ρ can be estimated as the processing time ratio between the CPU and the GPU for the same image. Fig. 4 shows the image partition strategy used in this work.

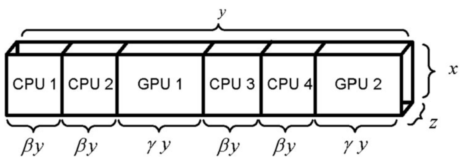

Fig. 4. CPU/GPU partition strategy supposing there are two GPUs and four CPUs.

Algorithm 1: Pseudocode of BME-LLSURE Implementation.

```
 1: % b denotes the number of spectral bands%
 2: % d denotes the number of OpenCL devices%
 3: Allocate memory in the device for $\mathbf{F}, \mathbf{S}$;
 4: Allocate memory in the device for $a$, $b$, $\sigma$, and $edgeImage$;
 5: Calculate $\mathbf{W}(\hat{\mathbf{B}})$, $\mathbf{V}(\hat{\mathbf{B}})$
 6: Create one context with d devices and b streams;
 7: Divide the image into d subimages, with $x \times \gamma y$ pixels
      for GPU and $x \times \beta y$ pixels for CPU
 8: for $i = 1 \rightarrow b$ do
 9:     Copy $i$-th band of the multispectral image from CPU
        to device in the $i$-th stream;
10:     Compute the estimate of band $\hat{\mathbf{B}}$ (BME kernel);
11:     Compute the mean and variance of image window $\omega_i$
        (LLSURE_sigma kernel);
12:     Apply weights $\lambda_i$ of (26) (LLSURE_lambda kernel);
13:     Normalize values to unsigned char (LLSURE_normal
        kernel);
14:     Apply (30) for edge-preserving (LLSURE_edges
        kernel);
15:     Read the $i$-th band from device;
16: end for
17: Subimages come together to form the output image;
18: Frees the memory allocated in device.
```

Finally, the pseudocode for implementing the BME-LLSURE algorithm using a single device is presented in Algorithm 1.

IV. Implementation and Performance Analysis

In this section, the implementation and the time performance analysis of the proposed architecture based on multiple devices computing are evaluated. These results are analyzed from the point of view of multispectral processing accuracy and parallel performance. Then, we describe the potential of the proposed multiple device parallel design. Two case studies are presented to prove the effectiveness of the proposal. The first is a synthetic image, which can be seen in Fig. 5(a). The second is a multispectral image collected by the UAV Hexacopter DJI F550 with Tetracam multispectral camera, presented in Fig. 5(d). Considering a 3×3 window, the GPU time is 0.459 s and CPU time is 1.12 s, the partition parameters are as follows: $n = m = 1$, $\rho = 2.44$, $\beta = 0.29$, $\gamma = 0.71$; however, for simplicity, in the case of three-band images, the CPU processes one band and GPU process two bands.

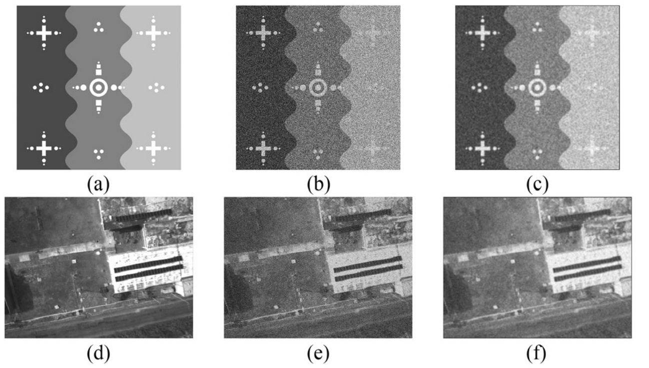

Fig. 5. Implementation results: Synthetic scene (a) original, (b) degraded, (c) enhanced; real multispectral image (d) original, (e) degraded, (f) enhanced.

A. Synthetic Image Data

The synthetic image (1024×1024 pixels) has elements and edges that present a challenge to the noise removal and enhancement. It was degraded, adding additive (Gaussian) and multiplicative (speckle) noise to simulate the physical effects of image capture with a flying device. The original and degraded image are shown in Fig. 5(a) and (b), respectively. The degraded image was processed with the proposed BME-LLSURE; see Fig. 5(c). In order to have a comparable metric of enhancement, in analogy to the image reconstruction, the quality metric defined as an improvement in the output signal-to-noise ratio (IOSNR) is employed as follows:

$$\mathrm{IOSNR} = 10\,\log_{10}\frac{\sum_{k=1}^{K}(\hat{b}_k - b_k)^2}{\sum_{k=1}^{K}(\hat{b}_k^{(BL)} - b_k)^2} \tag{34}$$

where b_k represents the value of the kth element (pixel) of the original image $\mathbf{B}$, $\hat{b}_k$ represents the value of the kth element (pixel) of the degraded image, and $\hat{b}_k^{(BL)}$ represents a value of the kth pixel of the image reconstructed using the proposed BME-LLSURE method. According to these quality metrics, the higher the IOSNR the better the improvement of the image enhancement/reconstructed with the employed algorithm.

B. Multispectral Image Data

The full dataset selected for the real case consists of multispectral images of 2048×1536 pixels, with three spectral bands corresponding to green, red, and NIR sensitivity, approximately equal to Landsat TM2, TM3, and TM4. We consider an aerial image of the School of Engineering, Autonomous University of Yucatán, Mérida, México. Fig. 5(d) shows the NIR band of the multispectral image, which is referred as UAV. Fig. 5(e) shows the degraded image using speckle and Gaussian noise. Fig. 5(f) shows the enhanced image using a 3×3 window for BME-LLSURE. From these figures, it can be seen that some structure is recovered and it is possible to see edges and objects that noisy image occludes.

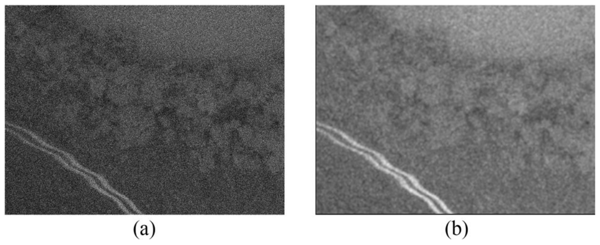

Fig. 6. Implementation results for UAV2: (a) multispectral image from cenote; (b) enhanced image with BME-LLSURE.

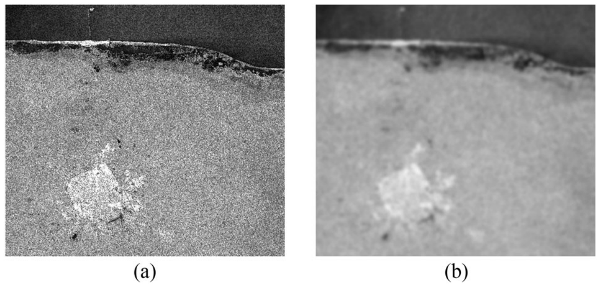

Fig. 7. Implementation results: (a) original Sentinel-1 scene [28]; (b) image enhanced applying the BME-regularized and edge-preserving LLSURE.

TABLE III
PROCESSING TIME (IN S) OF THE MBE-LLSURE IMPLEMENTATIONS

Scene	C++ 1 band	CPU* 1 band	GPU* 1 band	CPU+GPU* 3 bands
Synthetic (3 × 3)	.62 s	0.410 s	0.180 s	0.410 s
Synthetic (5 × 5)	1.36 s	0.464 s	0.281 s	0.464 s
UAV (3 × 3)	1.87 s	1.120 s	0.459 s	1.120 s
UAV (5 × 5)	4.14 s	1.376 s	0.496 s	1.376 s
UAV 2 (3 × 3)	1.92 s	1.145 s	0.465 s	1.145 s
UAV 2 (5 × 5)	4.25 s	1.462 s	0.513 s	1.462 s
Sentinel-1[28] (3 × 3)	212.6 s	34.01 s	21.12 s	–
Sentinel-1[28] (5 × 5)	483.9 s	65.28 s	42.42 s	–

*Implemented with OpenCL.

TABLE IV
IOSNR (DB) REGISTERED BY THE MBE-LLSURE IMPLEMENTATIONS

Scene	3 × 3 window	5 × 5 window	Scene	3 × 3 window	5 × 5 window
Synthetic	9.860	8.827			
UAV (1)	5.738	5.132	UAV 2 (1)	7.318	7.119
UAV (2)	10.406	8.539	UAV 2 (2)	3.951	2.079
UAV (3)	9.553	8.343	UAV 2 (3)	13.712	11.396

(1) Red band, (2) Green band, (3) NIR band.

Fig. 6 presents a cenote image captured with the UAV located at 21°13′39.52″N, 88°35′17.86″W, which is referred as UAV2. The UAV2 image was shot at 170 m off the ground with a resolution of 60 mm. Fig. 6(a) shows the cenote multispectral image, and Fig. 6(b) shows the resulting enhanced image via BME-LLSURE algorithm.

C. SAR Image Data

The second real case consists of a one-band Sentinel-1 [27] SAR image. SAR images suffer speckle noise, and they are suitable to be enhanced with the proposed implementation. Sentinel-1 is a two-satellite constellation with the prime objectives of land and sea monitoring. The goal of the mission is to provide C-band SAR data with medium- and high-resolution imaging in all weather conditions. The C-SAR is capable of obtaining night imagery and detecting small movement on the ground, which makes it useful for land and sea monitoring. The image used corresponds to the Yucatán Peninsula with 25206 × 15157 16-b pixels. The white area in the image corresponds to Mérida city located at 20°58′24.7″N, 89°37′14.2″W. The image occupies 764.2 MB, and the implementation demands as much as seven float point memory buffers resulting in approximately 10 GB; the GPU used has only 4 GB, hence it was necessary to split the image and send parts to GPU. Fig. 7(a) shows a selected region from Sentinel-1 original image; Fig. 7(b) shows enhanced image region using a 3 × 3 window for BME-LLSURE.

D. Performance Analysis

In this section, the time performance analysis of the multiple device implementation is presented. The algorithm has been tested on an iMac with two devices: CPU and GPU associated with the "Apple" platform. The CPU is an Intel Core i5 @ 3.4 GHz with 24 GB of RAM and four compute units; the GPU is an NVIDIA GeForce GTX 780M with 4 GB of memory and eight compute units, with 192 compute elements (cores) each.

Table III summarizes the overall time performance achieved with the proposed approach. The size of the window is a processing time factor, since there are more pixels to process for each window center. There are four enhanced images: one synthetic, two real multispectral image with three bands, and a single band Sentinel-1 image. Table III shows the processing times for CPU and GPU when only one band is enhanced; to process all bands, the value shown must be multiplied by three. The last column of Table III shows the time required to process all bands according to the spectral partition; the overall time is the greatest between the CPU time (one band) and the GPU time (two bands). Additionally, Table IV summarizes the resulting IOSNR metric for the enhanced image. There is a difference between using 3 × 3 or 5 × 5 windows for the method, with the result that enhanced images with the smaller window are better estimations of real images.

The implementation of the MBE-LLSURE algorithm using the proposed multiple device architecture and 3 × 3 windows takes only 1.120 s for the first multispectral RS image, in contrast to the 5.61 s (1.67 s × 3 bands) required with the C++ CPU-based implementation, with a significant speed up of 5×. When the size of the image grows, as in the Sentinel-1 image, the speed up of the GPU versus the CPU-based is 11.5×. The proposed implementation helps to drastically reduce the overall processing time of the algorithm. In regard to IOSNR, the contents of the image highly affects the result, since for a urban scenario (UAV data) the major recovery is in the red band; in the forest scenario (UAV2 data) the best results are achieved on the NIR band.

V. Conclusion

The principal result of this study relates the digital signal processing oriented solution of RS enhancing imaging problems (i.e., real large-scale multispectral images of 2048×1536 pixels, and 25206×15157 pixels with one band) by integrating parallel computing techniques with OpenCL devices. In the first stage, the BMR and the LLSURE techniques were algorithmically adapted for image enhancement and postprocessing edge-preserving denoising filtering. In the study, the enhancement approach was stated and addressed as an uncertain ill-posed inverse problem of nonparametric estimation of the SSP. This pattern is associated to unknown statistics of perturbations of the recorded data in the turbulent medium, finite dimensionality of measurements, and uncontrolled UAV vibrations. In a second stage, the large-scale image enhancement tasks are efficiently implemented using multiple devices with the aim to accelerate the highly demanding operations embedded in the analysis of a particular geographical region. The implementation has been tested with images acquired by a UAV; the obtained results show an enhancement based on the IOSNR metric. The parallel OpenCL-based approach significantly reduces the overall computational load of the algorithms obtaining a significant speed up of $11.5\times$ over a C++ CPU-based implementation when processing a high spatial resolution image, justifying the use of GPU units. Finally, through the results of the multispectral test scenarios, we have concluded that near real-time image processing can be achieved with the proposed approach. The proposed implementation is useful for applications where images are corrupted with speckle or white noise, since the implemented algorithms handle these very well. OpenCL is a very attractive option for parallelism because it is supported by different vendors and architectures; however, for RS images, it is necessary to pay special attention to issues, such as memory overflow, device capabilities, event synchronization, and others. Other problems arise because RS image sizes are usually not power of 2, or are too big to fit in GPU device memories, the proposed implementation address this issue by querying the device capabilities and adapting buffer and work sizes. Finally, technology provides new multiprocessor devices that are smaller, but more powerful, which lead us to think that our proposal would execute on board of UAV in the near future.

References

[1] F. M. Henderson and A. J. Lewis (Eds), *Principles and Applications of Imaging Radar, Manual of Remote Sensing, vol. 2*, 3rd ed. New York, NY, USA: Wiley, 1998.

[2] J. W. Van Wagtendonk, R. R. Root, and C. H. Key, "Comparison of AVIRIS and landsat ETM+ detection capabilities for burn severity," *Remote Sens. Environ.*, vol. 92, no. 3, pp. 397–408, 2004.

[3] C.-I. Chang, *Hyperspectral Data Exploitation: Theory and Applications*. New York, NY, USA: Wiley-Interscience, 2007.

[4] A. Castillo, D. Torres, and Y. Shkvarko, "Experiment design regularization-based hardware/software codesign for real-time enhanced imaging in uncertain remote sensing environment," *EURASIP J. Adv. Signal Process.*, vol. 2010, 2010, Art. no. 10.

[5] T. Qiu, A. Wang, N. Yu, and A. Song, "LLSURE: Local linear SURE-based edge-preserving image filtering," *IEEE Trans. Image Process.*, vol. 22, no. 1, pp. 80–90, Jan. 2013.

[6] P. Jain and V. Tyagi, "A survey of edge-preserving image denoising methods," *Inf. Syst. Frontiers*, vol. 18, no. 1, pp. 159–170, Feb. 2016.

[7] C. A. Lee, S. D. Gasster, A. Plaza, C. I. Chang, and B. Huang, "Recent developments in high performance computing for remote sensing: A review," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 4, no. 3, pp. 508–527, Sep. 2011.

[8] S. S. Maddikonda and S. S. G. A, "SAR image processing using GPU," in *Proc. Int. Conf. Commun. Signal Process.*, 2014, pp. 448–452.

[9] A. Castillo, R. Carrasco, J. Ortegón, and J. Vázquez, "A new tool for intelligent parallel processing of radar/SAR remotely sensed imagery," *Math. Problems Eng.*, vol. 2013, pp. 1–10, 2013.

[10] C. Gonzalez, S. Sanchez, A. Paz, J. Resano, D. Mozos, and A. Plaza, "Use of FPGA or GPU-based architectures for remotely sensed hyperspectral image processing," *Integr., VLSI J.*, vol. 46, pp. 89–103, 2013.

[11] E. Christophe, J. Michel, and J. Inglada, "Remote sensing processing: From multicore to GPU," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 4, no. 3, pp. 643–652, Sep. 2011.

[12] G. Bilotta, R. Z. Sanchez, and G. Ganci, "Optimizing satellite monitoring of volcanic areas through GPUs and multi-core CPUs image processing: An openCL case study," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 6, no. 6, pp. 2445–2452, Dec. 2013.

[13] F. Huang, S. Bu, J. Tao, and X. Tan, "OpenCL implementation of a parallel universal kriging algorithm for massive spatial data interpolation on heterogeneous systems," *ISPRS Int. J. Geo-Inf.*, vol. 5, no. 6, p. 96, 2016.

[14] G. M. Callicó, S. Lopez, B. Aguilar, J. F. López, and R. Sarmiento, "Parallel implementation of the modified vertex component analysis algorithm for hyperspectral unmixing using opencl," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 7, no. 8, pp. 3650–3659, Aug. 2014.

[15] F. Huang, J. Zhou, J. Tao, X. Tan, S. Liang, and J. Cheng, "Pmodtran: A parallel implementation based on modtran for massive remote sensing data processing," *Int. J. Digital Earth*, vol. 9, no. 9, pp. 819–834, 2016.

[16] A. Castillo, Y. Shkvarko, D. Torres, and H. Perez, "Convex regularization-based hardware/software co-design for real-time enhancement of remote sensing imagery," *J. Real-Time Image Process.*, vol. 4, no. 3, pp. 261–272, 2009.

[17] I. E. Villalon-Turrubiates, "Remote sensing signatures extraction for hydrological resources management applications," in *Proc. IEEE/ACS Int. Conf. Comput. Syst. Appl.*, 2009, pp. 567–570.

[18] Y. Shkvarko, H. Perez, and A. Castillo, "Enhanced radar imaging in uncertain environment: A descriptive experiment design regularization paradigm," *Int. J. Navigat. Observ*, vol. 8, p. 11, 2008.

[19] I. Villalon-Turrubiates and M. Llovera-Torres, "Archaeological land use characterization using multispectral remote sensing data," in *Proc. IEEE Int. Geosci. Remote Sens. Symp.*, 2011, pp. 86–89.

[20] Y. V. Shkvarko, "Unifying experiment design and convex regularization techniques for enhanced imaging with uncertain remote sensing data—part I: Theory," *IEEE Trans. Geosci. Remote Sens.*, vol. 48, no. 1, pp. 82–95, Jan. 2010.

[21] T. Inc., "Tetracam -ADC lite," (2016). [Online]. http://www.tetracam.com/Products-ADC_Lite.htm

[22] A. N. Tikhonov and V. Y. Arsenin, *Solution of Ill-Posed Problems*. New York, NY, USA: Winston, 1977.

[23] R. Tibshirani, "Stein's Unbiased Risk Estimate." [Online]. Available: http://www.stat.cmu.edu/∼larry/=sml/stein.pdf. Accessed on: Feb. 24, 2016.

[24] K. Group, "OpenCL - the open standard for parallel programming of heterogeneous systems." (2016). [Online]. https://www.khronos.org/opencl/

[25] D. B. Kirk and W. H. Wen-mei, *Programming Massively Parallel Processors: A Hands-on Approach*. Burlington, MA, USA: Morgan Kaufmann, 2010.

[26] J. Sanders and E. Kandrot, *CUDA by Example: An Introduction to General-Purpose GPU Programming*. Boston, MA, USA: Addison-Wesley Prof., 2011.

[27] K. Fletcher and European Space Agency, Eds., *Sentinel-1: ESA's Radar Observatory Mission for GMES Operational Services*, ser. SENTINEL. Noordwijk, the Netherlands: ESA Communications, 2012.

[28] "EO Data—Earth Online—ESA." (2016). [Online]. https://earth.esa.int/