Chapter 3

Development of Remote Sensing Single Image Super-resolution using GPU-Accelerated Adaptive Dictionary Learning and Sparse Representations

3.1 Introduction

SR is highly useful in multispectral (MS) RS images because it can enhance the spatial resolution of the image, allowing for better identification and classification of features on the ground. It is highly significant in applications such as land cover mapping, crop monitoring, and urban planning, where the ability to distinguish fine image details is crucial for accurate analysis and decision-making. Additionally, SR can improve the overall quality of the image, making it easier to interpret and extract meaningful information. Learning-based techniques, especially sparse representation-based dictionary learning techniques, have gained significant improvements in RS super-resolution because these techniques allow the representation of MS image data using only a few of basis functions or columns (atoms) of the dictionary, which can be learned adaptively from the data itself. By decomposing the data into a sparse linear combination of these basis functions, these techniques can effectively capture the underlying structure and variability in the data, making them well-suited for SR of MS images.

As discussed in chapter 2, the dictionary learning techniques can be divided into two categories depending upon the number of images used in the training set: global dictionary learning and adaptive dictionary learning. Adaptive dictionary learning techniques are more beneficial than global dictionary in MS remote sensing super-resolution because they can capture the texture and structure details of the image better. Global dictionaries, which are constructed based on a large collection of training images, may not be able to effectively represent the unique features and structures present in a given MS image. Adaptive dictionary learning techniques, on the other hand, can adapt to the specific content of the image being processed, resulting in more accurate and efficient SR. By learning a dictionary tailored to the image, these techniques can capture the underlying structures and patterns at a finer scale, resulting in more detailed and accurately reconstructed images. Additionally, adaptive dictionary learning can be more computationally efficient, as they only need to learn the dictionary from the specific image being processed, rather than a large collection of training images.

Feature extraction is a critical step in achieving high-quality SR using sparse representations and dictionary learning, because it helps to identify the relevant information in LR images that can be used to enhance the resolution. By extracting meaningful features, the high-frequency information that is lost in the downsampling process can be recovered more effectively, resulting in a more accurate and visually pleasing HR image. Feature extraction also helps to improve the efficiency of the sparse coding process, making it easier to find the best representation of the LR image in terms of the learned dictionary.

Sparse representations and dictionary learning-based image SR algorithms are computationally intensive and require a significant amount of processing power to achieve high-quality results, especially for RS images having large image sizes. The sparse representation and dictionary training are to be carried out on small overlapping image patches extracted from the LR image; several thousands of patch-based regularization sub-problems need to be solved sequentially for sparse representation of the whole image. A trained HR dictionary is used to represent a patch of the target HR image in terms of a sparse coefficients. The coefficients are obtained by solving a highly non-linear optimization problem, which can be time-consuming, especially when the size of the image is large. The dictionary learning process also involves solving many optimization problems, which are computationally expensive. Moreover, MS remote sensing images consist of several spectral bands leading to a big data. Therefore, it is computationally exhaustive to restore HR multispectral images from the LR multispectral bands. In order to circumvent these problems, parallel computing approach can be exploited to design highly parallelized algorithms, which can be implemented using GP-GPU for real-time SR of RS images. CUDA-based GPU implementation can speed up these computations significantly by allowing for the parallel processing of matrix operations and other linear algebraic operations involved in these algorithms. By taking advantage of the thousands of parallel cores on a GPU, it is possible to achieve significant speedups over a CPU implementation. In particular, the CUDA-enabled GPU is the best choice to handle this computational burden effectively.

In this chapter, we develop a CUDA-GPU accelerated SISR method for MS remote sensing images using adaptive dictionary learning and sparse representations. In many sparse coding-based SR methods, gradient-based feature extraction is used in order to retrieve significant edges, but it fails to produce sharp edges. In the proposed work, a new feature extraction scheme based on the combination of Sobel filters in horizontal and vertical directions, difference of Gaussians (DoG) and fast Fourier transform (FFT) filters are used to restore the sharp edges significantly. For real-time applications, hybrid CPU-GPU algorithms based on CUDA programming model for dictionary learning and sparse reconstruction using hardware acceleration with GPU hardware are proposed.

3.1.1 Main contributions of the chapter

The main contributions of the chapter are summarized as follows:

(i) Developed a novel framework for MS remote sensing image SR using sparse coding and the adaptive dictionary learning method. Instead of training from external HR images, coupled dictionaries based on sparse representations are learnt from the given LR multispectral remote sensing image itself. A new feature extraction scheme based on DoG, Sobel and FFT filters is introduced for efficient dictionary learning as well as reconstruction of HR images.

(ii) Proposed fast adaptive dictionary learning based on K-singular value decomposition (K-SVD) and a sparse reconstruction algorithm; hardware acceleration with NVIDIA P100 GP-GPU hardware is achieved using the CUDA programming model. The parallel framework for SR of MS remote sensing image can process real MS remote sensing images up to 2048×2048 within a few seconds for different upscaling factors.

(iii) Simulations are carried out using real MS remote sensing images captured by the Indian Satellite with Linear Imaging Self Scanner (LISS-IV) sensor and procured from NRSC, ISRO. Results are evaluated in terms of visual quality and objective criteria, besides the computation time.

The rest of the chapter is organized as follows: Section 3.2 explains some prior art on the SISR model based on dictionary learning and sparse representations. The proposed sparse representation-based SR method for RS images and its implementation on CUDA are detailed in Section 3.3. Experimental results and different evaluation parameters for comparisons are demonstrated in section 3.3. Finally, Section 3.5 concludes the paper.

3.2 Prior Art

3.2.1 Sparse representation model for super-resolution

In SISR model, the observed LR image $\mathbf{Y}$ is obtained from the HR image $\mathbf{X}$ based on following reconstruction constraint, as follows:

$$\mathbf{Y} = SH\mathbf{X} \tag{3.1}$$

Here H and S represent the blurring and down-sampling operators, respectively. Recovering the HR image $\mathbf{X}$ from the LR image is an ill-posed problem as many HR images $\mathbf{X}$ may obey the above equation, which can be accurately solved as an optimization problem by introducing the knowledge of sparsity of the image over any transform or dictionary as a priori constraint. In practice, due to the limitation of the size of the dictionary required for the sparse representation (coding) of the whole image, the main optimization problem is split into several patch-based subproblems. An image patch can be represented by sparse linear combination of elements (atoms) from an appropriately chosen over-complete dictionary. In order to reconstruct the HR image patches $\mathbf{x}$, the sparse co-efficients can be obtained by using an LR dictionary ($\mathbf{D}_\ell$) learned from the LR image patches $\mathbf{y}$ taken from the LR input image $\mathbf{Y}$. In particular, the following minimization problem can be defined to compute the sparse-co-efficients vector $\boldsymbol{\alpha}$:

$$\min_{\boldsymbol{\alpha}} \|\boldsymbol{\alpha}\|_1 \text{ subject to } \|\mathbf{D}_\ell\boldsymbol{\alpha} - \mathbf{y}\|_2^2 \leq \epsilon, \tag{3.2}$$

An unconstrained version of the above equation may be written as follows:

$$\min_{\boldsymbol{\alpha}} \|\mathbf{D}_\ell\boldsymbol{\alpha} - \mathbf{y}\|_2^2 + \lambda\|\boldsymbol{\alpha}\|_1, \tag{3.3}$$

where the regularization parameter λ is used to trade-off between sparsity and accuracy of the output. This optimization problem can be efficiently solved by using the fast ℓ_1-minimization algorithms such as LASSO [27], ℓ_1-feature-sign [118], etc. Since individual LR and HR patch pairs share the same sparse coefficients vector, finally, the desired HR image patch $\mathbf{x}$ can be obtained by:

$$\mathbf{x} = \mathbf{D}_h\boldsymbol{\alpha} \tag{3.4}$$

Once all the HR patches are reconstructed by the above process, they are tiled (stitched) together to get an approximate HR image $\hat{\mathbf{X}}$, which may be further refined by solving another optimization problem with the global penalty term in Eq. 3.1, as discussed in Chapter 1 (Eq. 1.13).

3.2.2 Dictionary learning

Dictionary learning is another key consideration in sparse representation. In order to learn the coupled dictionary $\mathbf{D}_c = [\mathbf{D}_h; \mathbf{D}_\ell]$, assume that there are N spatially correlated HR samples denoted by $\mathbf{X}_h = [\mathbf{x}_{h1}, \mathbf{x}_{h2}, \ldots, \mathbf{x}_{hN}]$ and N LR samples denoted by $\mathbf{Y}_l = [\mathbf{y}_{l1}, \mathbf{y}_{l2}, \ldots, \mathbf{y}_{lN}]$. Since $\mathbf{X}_h$ and $\mathbf{Y}_l$ would share a common sparse coefficients matrix, a joint dictionary training approach can be adopted by enforcing the sparsity $\|\mathbf{Z}\|_1$ on the concatenated data $\mathbf{Y}_c = [\mathbf{X}_h; \mathbf{Y}_l]$ using the following optimization problem:

$$\{\mathbf{D}_c, \mathbf{Z}^*\} = arg \min_{\{\mathbf{D}_c, \mathbf{Z}\}} \|\mathbf{Y}_c - \mathbf{D}_c\mathbf{Z}\|_F^2, \text{ subject to} \quad \|\boldsymbol{\alpha}_j\|_0 \leq T \tag{3.5}$$

where $\boldsymbol{\alpha}_j$ is the j^{th} column vector in $\mathbf{Z}$ and T indicates the sparsity level. Detailed discussion on dictionary training is already done in Chapter 1 (Subsection 1.3.3).

3.3 Materials and Methods

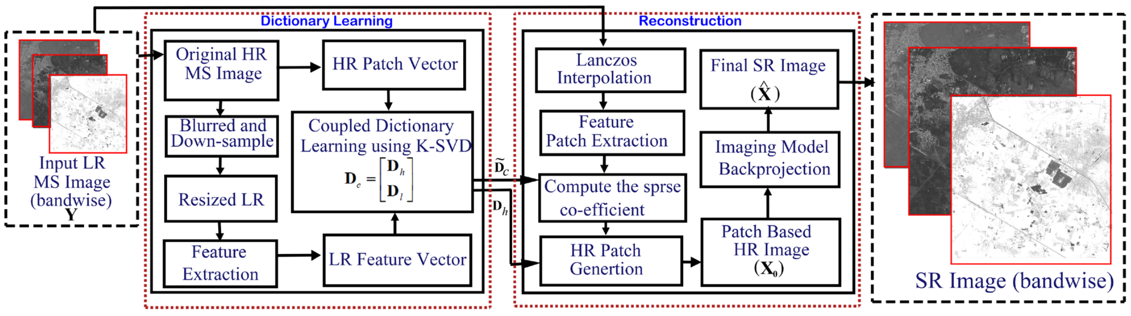

Dictionary Learning
Reconstruction
Input LR MS Image (bandwise) Y
Original HR MS Image
HR Patch Vector
Blurred and Down-sample
Coupled Dictionary Learning using K-SVD
$\mathbf{D}_c = \begin{bmatrix}\mathbf{D}_h \\ \mathbf{D}_l\end{bmatrix}$
Resized LR
Feature Extraction
LR Feature Vector
$\tilde{\mathbf{D}}_c$
$\mathbf{D}_h$
Lanczos Interpolation
Feature Patch Extraction
Compute the sprse co-efficient
HR Patch Genertion
Final SR Image ($\hat{\mathbf{X}}$)
Imaging Model Backprojection
Patch Based HR Image ($\mathbf{X}_0$)
SR Image (bandwise)

Figure 3.1: Block diagram of the proposed RS SISR.

The proposed adaptive dictionary learning-based MS image super-resolution framework is shown in Figure 3.1. It is assumed that many similar image patches may exist in the same LR image, both for the same and across different scales. Therefore, to make the dictionary learning method more effective and accurate for RS applications, adaptive dictionary learning is used, where the LR/HR dictionaries have been learned and updated directly from overlapping patches of the given

LR multispectral image. Once the dictionaries are trained, the SR multispectral image is reconstructed from its LR multispectral image by solving the sparse coding problem. The proposed framework processes each MS remote sensing image band separately.

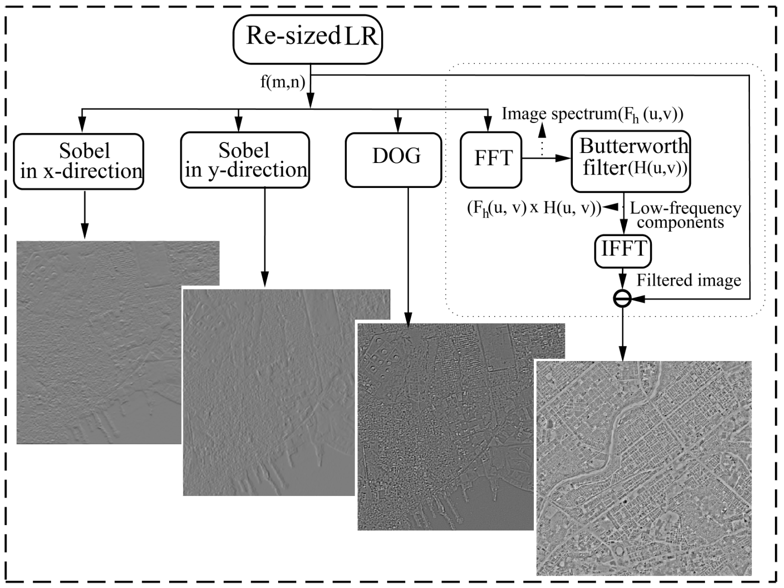


Figure 3.2: Feature extraction step.

3.3.1 CPU implementation of MS remote sensing image SR based on adaptive dictionary learning

The adaptive dictionary learning-based MS remote sensing image SR consists of the following:

3.3.1.1 Feature extraction

In order to mimic the process that any LR image would be going through during its acquisition, and subsequently undoing it to achieve the corresponding HR image, we carry out the following two steps. First, the original LR multispectral band image is blurred and downsampled. Next, Lanczos interpolation is done on the blurred and downsampled image to obtain the re-sized LR or intermediate HR image equal in size with the actual LR image. We apply feature extraction step on the intermediate HR image to extract the high-frequency information from it in order to improve the sparse representation accuracy. The limitation of the feature extraction based on first- and second-order gradient filters [118] is that significant edges are not restored, resulting in smooth edges. To address this limitation, a new feature extraction method is used, which can easily restore sharp edges and high-frequency features from the LR image. In order to reconstruct an image with sharp edges, difference of Gaussians (DoG) and Sobel filters in horizontal and vertical directions are applied on the re-sized LR. Generally, 2D Gaussian function i.e. the distribution of uncorrelated random variables i, j (representing the horizontal and vertical directions) having a bivariate normal distribution and equal standard deviation, can be expressed as:

$$G(i,j,\sigma) = \frac{1}{2\pi\sigma^2} e^{\frac{i^2+j^2}{2\sigma^2}}, \tag{3.6}$$

Similarly, the DoG filter can be described by:

$$\mathrm{DoG}(i,j) = G(i,j,\sigma_1) - G(i,j,\sigma_2). \tag{3.7}$$

Now, this DoG filter is applied on resized LR with $\sigma_1 = 1.0$ and $\sigma_2 = 1.6$. Next, the sobel operator is applied on re-sized LR to detect gradient maps, given by:

$$S_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, S_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}, \tag{3.8}$$

where S_x and S_y represent the horizontal and vertical gradients, respectively. Sobel and DoG filters are first- and second-order derivative filters, respectively that are applied to each LR image separately to obtain feature maps containing various features, like edges in horizontal and vertical directions, corners, and blobs.

To retrieve high-frequency information, FFT is also performed on re-sized LR, which can be expressed as $\{(x,y)|x \in (0,M); y \in (0,N)\}$; to generate an image spectrum. The image spectrum $F(u, v)$ of re-sized LR can be described as:

$$F(u,\ v) = \frac{1}{\sqrt{MN}} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y)\, e^{-j2\pi\left[\frac{ux}{M}+\frac{vy}{N}\right]},$$
$$u = [0, 1, ..., M-1]; v = [0, 1, ..., N-1] \tag{3.9}$$

A Butterworth low-pass filter is used to extract the low frequency component from an image spectrum with the center at $(u_c, v_c) = (M/2, N/2)$, size $M \times N$. This filter has the following transfer function:

$$H(u,\ v) = \frac{1}{1 + \left[\frac{w_0}{w_c(u,v)}\right]^{2n}}, \tag{3.10}$$

Here $w_c(u, v)$ is measured by the distance between a point on the spectrum (u, v) and the center (u_c, v_c); w_0 represents the cut-off frequency and n=2. Now, this Butterworth filter is subsequently convolved with $F(u, v)$ in order to extract the low frequency component $F_h(u, v)$ from $F(u, v)$, as follows:

$$F_h(u,\ v) = F(u,\ v) * H(u,\ v), \tag{3.11}$$

In order to obtain the filtered image $f_h(x, y)$, inverse FFT (IFFT) is applied on $F_h(u,\ v)$:

$$f(u,\ v) = \frac{1}{\sqrt{MN}} \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} F_h(x, y)\, e^{j2\pi\left[\frac{ux}{M}+\frac{vy}{N}\right]} \tag{3.12}$$

Finally, high-frequency information from the re-sized LR image are extracted by subtracting the FFT-filter output $(f(u,\ v)$ from the re-sized LR image. The feature extraction procedure is explained graphically in Fig. 3.2.

3.3.1.2 K-SVD dictionary training

In order to learn a coupled overcomplete dictionary $\mathbf{D}_c$, LR training feature patch matrix $\mathbf{Y}_\ell \in \mathbb{R}^{n^l \times P} = \{\mathbf{y}_{\ell 1}, \mathbf{y}_{\ell 2}, ..., \mathbf{y}_{\ell P}\}$, are obtained from the feature extraction step, and HR training patch matrix $\mathbf{X}_h \in \mathbb{R}^{n^h \times P} = \{\mathbf{x}_{h1}, \mathbf{x}_{h2}, ..., \mathbf{x}_{hP}\}$, are extracted from the original input image $\mathbf{X}$; n^h and n^l representing the lengths of each HR and LR feature patch vectors, respectively. Each LR feature patch vector is formed by concatenating results of four filters in the feature extraction performed on the interpolated LR image. For learning LR dictionary $\mathbf{D}_\ell$, we extract feature patches from the interpolated image and obtain the feature patch matrix $\mathbf{Y}_\ell$ as mentioned above. Therefore, the size of each LR feature vector, n^l is $4n^h \times 1$. Next, a combined HR-LR patch dataset $\mathbf{Y}_c$ is created by concatenating $\mathbf{X}_h$ and $\mathbf{Y}_\ell$, i.e. $\mathbf{Y}_c = [\mathbf{X}_h; \mathbf{Y}_\ell] \in \mathbb{R}^{(n^h+n^l)\times P}$. The above sample patches are also used for initializing $\mathbf{D}_\ell$ and $\mathbf{D}_h$ in our experiments. We have not used the random Gaussian or DCT matrices to initialize the dictionary for simplicity.

Algorithm 1: K-SVD algorithm for dictionary learning.

```
   Data:
         $\mathbf{Y}_c$: Patch matrix
         $\mathbf{D}_0$: initial dictionary
         $T$: target sparsity
         $N$: number of iterations
   Result: Dictionary $\mathbf{D}_c$ and sparse matrix $\boldsymbol{\alpha}$ such that $\mathbf{Y}_c \approx \mathbf{D}_c\boldsymbol{\alpha}$
 1 Initialization: Set $\mathbf{D}_c := \mathbf{D}_0$ ;
 2 Set J=1 ;
 3 while not converge do
 4     Sparse coding stage:
 5     for $i \leftarrow 1$ to $P$ do
 6         Solve $\alpha_i := OMP(\mathbf{Y}_c, \mathbf{D}_c, K)$
           $\boldsymbol{\alpha}_i : \underset{\alpha_i}{\operatorname{argmin}} \|\mathbf{y}_i - \mathbf{D}_c\boldsymbol{\alpha}_i\|_2^2$ subject to $\|\boldsymbol{\alpha}_i\|_1 \leq T$ ;
 7     end
 8     Dictionary update stage:
 9     for $k \leftarrow 1$ to $K$ do
10         $I \leftarrow \{j|\boldsymbol{\alpha}_{j,k} \neq 0\}$ ;
11         Error matrix calculation: $\mathbf{E}_k = \mathbf{Y}_c - \sum_{j\neq k} \mathbf{d}_j\boldsymbol{\alpha}_I^j$ ;
12         Using SVD decomposition: $\mathbf{E}_k = \mathbf{U}\boldsymbol{\Lambda}\mathbf{V}^T$ ;
13         Updating dictionary column: $\mathbf{d}_k \leftarrow \mathbf{U}(:,1)$;
14         Updating sparse vector: $\boldsymbol{\alpha}_k = \mathbf{V}(:,1) * \boldsymbol{\Lambda}(1,1)$
15     end
16     Set $J = J + 1$
17 end
```

The HR-LR dictionary pairs $\mathbf{D}_h \in \mathbb{R}^{n^h\times K}$ and $\mathbf{D}_\ell \in \mathbb{R}^{n^l\times K}$ are jointly learned in the form of $\mathbf{D}_c \in \mathbb{R}^{(n^h+n^l)\times K}$ using $\mathbf{Y}_c$. In principle, $\mathbf{X}_h$ and $\mathbf{Y}_\ell$ have similar sparse representation vectors, therefore, by enforcing a common sparsity term $\|\mathbf{Z}\|_1$ both for LR and HR patches, a joint dictionary $\mathbf{D}_c$ can be trained by solving the following composite optimization problem:

$$\min_{\{\mathbf{D}_h,\mathbf{D}_\ell,\mathbf{Z}\}} \|\mathbf{Y}_c - \mathbf{D}_c\mathbf{Z}\|_F^2 + \lambda\|\mathbf{Z}\|_1, \tag{3.13}$$

where

$$\mathbf{D}_c = \begin{bmatrix} \frac{1}{\sqrt{\mathrm{n^h}}}\mathbf{D}_h \\ \frac{1}{\sqrt{\mathrm{n^l}}}\mathbf{D}_\ell \end{bmatrix} \quad , \quad \mathbf{Y}_c = \begin{bmatrix} \frac{1}{\sqrt{\mathrm{n^h}}}\mathbf{X}_h \\ \frac{1}{\sqrt{\mathrm{n^l}}}\mathbf{Y}_\ell \end{bmatrix},$$

and λ is the Lagrange multiplier used for balancing the sparsity and fidelity of the approximation. The K-SVD dictionary learning technique [80], as shown in Algorithm 1 is used to solve Eq. 3.13 approximately. The K-SVD performs two steps iteratively: firstly, a sparse coding of $\mathbf{Y}_c$ is computed using the OMP by fixing the dictionary. Secondly, atom-wise dictionary updating is done iteratively to simplify the updating step.

3.3.1.3 Sparse Reconstruction

In the reconstruction phase, the input LR multispectral $\mathbf{Y}$ image having L bands is super-revolved band-wise. Each band image is upsampled by Lanczos interpolation to the required size and then the upsampled image is passed through the feature extraction step in order to extract four high-frequency feature maps of equal size and then obtained feature patches of size $\sqrt{n^\ell} \times \sqrt{n^\ell}$ (n^ℓ: the number of pixels in a patch) with single pixel overlap from each feature map. Finally, four feature patches from a particular pixel location are vectorized and concatenated to form a single feature vector $\mathbf{y}_{\ell_i}$ of dimension $4n^\ell \times 1$. In a band, every LR multispectral patch $\mathbf{y}_{\ell_i}$ is sparsely represented by $\boldsymbol{\alpha}_i$ using the LR multispectral dictionary $\mathbf{D}_c$. The process is repeated for the remaining bands as well. Mathematically, given $\mathbf{D}_c$ and the feature patch $\mathbf{y}_{\ell_i}$, the following minimization problem is solved using the feature sign method to obtain $\boldsymbol{\alpha}_i$ [118]:

$$\boldsymbol{\alpha}_i = \min_{\boldsymbol{\alpha}} \|\tilde{\mathbf{D}}_c\boldsymbol{\alpha}_i - \tilde{\mathbf{y}}_{\ell_i}\|_2^2 + \lambda\|\boldsymbol{\alpha}_i\|_1, \tag{3.14}$$

where $\tilde{\mathbf{D}}_c = \begin{bmatrix} \mathbf{D}_\ell \\ E\mathbf{D}_h \end{bmatrix}$, E extracts the region of overlaps between the target patch and already reconstructed HR patches, and $\tilde{\mathbf{y}}_{\ell_i} = \begin{bmatrix} \mathbf{y}_{\ell_i} \\ \mathbf{g} \end{bmatrix}$, g represents pixels in the previously reconstructed HR patch in the overlapped region. Since, individual HR and LR patches share the same sparse representation coefficients for $\mathbf{D}_c$, the target HR MS image patch $\mathbf{x}_{h_i}$ is obtained by multiplying the sparse coefficients $\boldsymbol{\alpha}_i$ with the HR dictionary $\mathbf{D}_h$ as follows:

$$\mathbf{x}_{h_i} = \mathbf{D}_h\boldsymbol{\alpha}_i, \tag{3.15}$$

Finally, an approximate HR image $\mathbf{X}_0$ is then reconstructed by tiling of all the HR image patches. By applying a back projection step in order to match the image degradation model, we can achieve the target HR MS image $\hat{\mathbf{X}}$ by solving the following quadratic optimization problem having a close-form solution, i.e.:

$$\hat{\mathbf{X}} = \operatorname*{argmin}_{\mathbf{X}} \|SH\mathbf{X} - \mathbf{Y}\|_2^2 + C\|\mathbf{X} - \mathbf{X}_0\|_2^2, \tag{3.16}$$

where C is the regularization parameter. The closed-form solution of Eq. 3.16 is:

$$\mathbf{X} = (S^T H^T SH + C)^{-1}(S^T H^T \mathbf{Y} + C\mathbf{X}_0) \tag{3.17}$$

However, gradient descent method is used to solve the optimization problem of Eq. 3.16 efficiently. The iterative update formula for this method is as follows:

$$\mathbf{X}_{k+1} = \mathbf{X}_k + \nu[S^T H^T(\mathbf{Y} - SH\mathbf{X}_k) + C(\mathbf{X} - \mathbf{X}_0)] \tag{3.18}$$

where $\mathbf{X}_k$ represents the HR image estimate at the k-th iteration, while ν denotes the gradient descent step size. The complete algorithm for the proposed method is presented in Algorithm 2.

Algorithm 2: Adaptive dictionary learning and sparse representation-based SISR algorithm

```
   Input: LR input image $\mathbf{Y}$
1  Apply Lanczos interpolation on $\mathbf{Y}$ to obtain re-sized $\mathbf{Y}$
2  Feature extraction step: Extract four high-frequency feature maps from
   re-sized $\mathbf{Y}$: Sobel in x–, and y–directions, DoG, Butterworth.
3  LR training feature patch matrix $\mathbf{Y}_\ell \in \mathbb{R}^{n^l \times P} = \{\mathbf{y}_{\ell 1}, \mathbf{y}_{\ell 2}, ...., \mathbf{y}_{\ell P}\}$, are
   obtained from the feature extraction step
4  HR training patch matrix $\mathbf{X}_h \in \mathbb{R}^{n^h \times P} = \{\mathbf{x}_{h1}, \mathbf{x}_{h2}, ...., \mathbf{x}_{hP}\}$, are extracted
   from re-sized $\mathbf{Y}$.
5  Dictionary construction: The HR-LR dictionary pairs $\mathbf{D}_h \in \mathbb{R}^{n^h \times K}$ and
   $\mathbf{D}_\ell \in \mathbb{R}^{n^l \times K}$ are jointly learned using K-SVD by solving Eq. using $\mathbf{Y}_c$
6  for each LR patch $\mathbf{y}_{\ell_i}$ of $\mathbf{Y}$ do
7      Solve the optimization problem with $\tilde{\mathbf{D}}_c$ and $\tilde{\mathbf{y}}_{\ell_i}$ defined in:
       $\boldsymbol{\alpha}_i = \min_{\boldsymbol{\alpha}} \|\tilde{\mathbf{D}}_c \boldsymbol{\alpha}_i - \tilde{\mathbf{y}}_{\ell_i}\|_2^2 + \lambda \|\boldsymbol{\alpha}_i\|_1$
8      Generate HR image patch $\mathbf{x}_{h_i} = \mathbf{D}_h \boldsymbol{\alpha}_i$. Tile all the HR image patches $\mathbf{x}_{h_i}$
       into $\mathbf{X}_0$
9  end
10 Apply global imaging constraint: $\hat{\mathbf{X}} = \text{argmin}_{\mathbf{X}} \|SH\mathbf{X} - \mathbf{Y}\|_2^2 + C\|\mathbf{X} - \mathbf{X}_0\|_2^2$
   Output: $\hat{\mathbf{X}}$
```

3.3.2 CUDA-GPU implementation

The proposed method is implemented on CUDA-enabled GPU hardware by exploiting the dedicated cuBLAS library and thread-level parallism. The NVIDIA cuBLAS is a dedicated GPU library for the basic linear algebra subroutines (BLAS). This library gives the user the access to the computational resources of a GPU.

3.3.2.1 CUDA GPGPU-based implementation of dictionary learning

The time-expensive iterative process in the K-SVD method is the sparse coding by the OMP [76]. As OMP consists of many matrix/vector operations, such as the matrix inverse, matrix-vector multiplication, and least-square approximation, it is very much suitable for parallel implementation using GPU. Before the standard OMP steps could be executed on GPU, the trained dictionary $\mathbf{D}_c$ and patch vectors $\mathbf{y}_i$ are

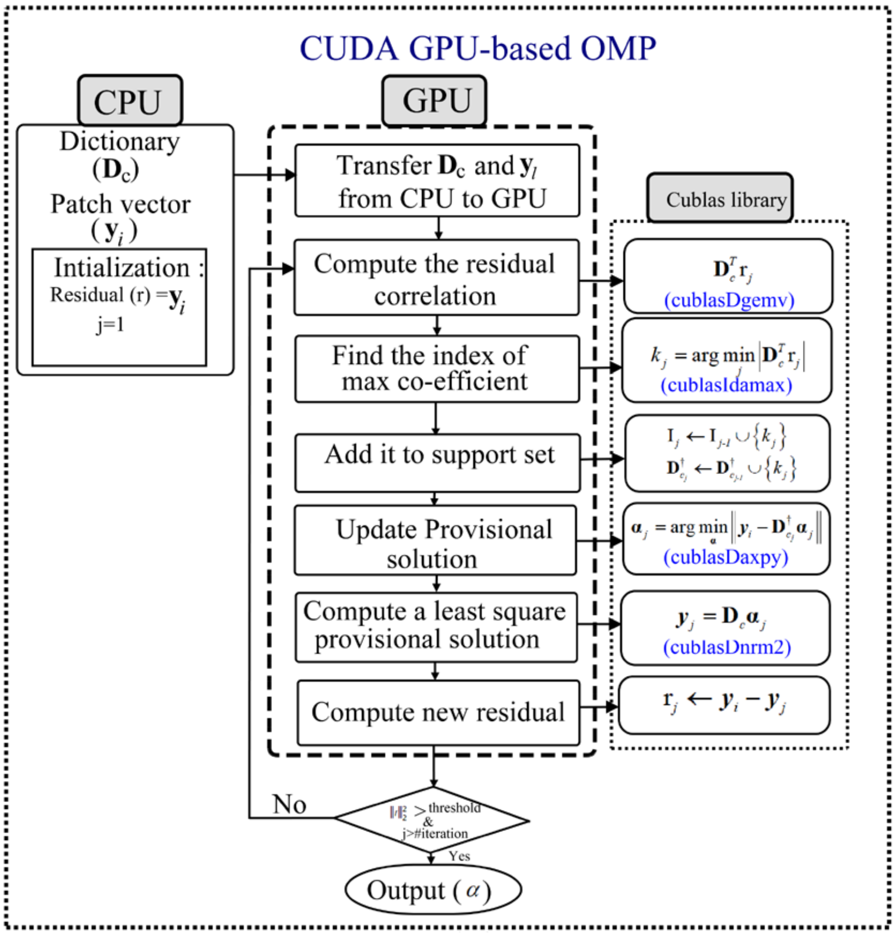

CUDA GPU-based OMP
CPU
GPU
Dictionary ($\mathbf{D}_c$)
Patch vector ($\mathbf{y}_i$)
Intialization : Residual (r) = $\mathbf{y}_i$ j=1
Transfer $\mathbf{D}_c$ and $\mathbf{y}_i$ from CPU to GPU
Compute the residual correlation
Find the index of max co-efficient
Add it to support set
Update Provisional solution
Compute a least square provisional solution
Compute new residual
Cublas library
$\mathbf{D}_c^T\mathbf{r}_j$ (cublasDgemv)
$k_j = \arg\min_j |\mathbf{D}_c^T\mathbf{r}_j|$ (cublasIdamax)
$\mathbf{I}_j \leftarrow \mathbf{I}_{j-1} \cup \{k_j\}$
$\mathbf{D}_{c_j}^\dagger \leftarrow \mathbf{D}_{c_{j-1}}^\dagger \cup \{k_j\}$
$\alpha_j = \arg\min_\alpha \|y_i - \mathbf{D}_{c_j}^\dagger \alpha_j\|$ (cublasDaxpy)
$y_j = \mathbf{D}_c \alpha_j$ (cublasDnrm2)
$\mathbf{r}_j \leftarrow y_i - y_j$
No
$\|\mathbf{r}\|_2^2$ > threshold & j>#iteration
Yes
Output (α)

Figure 3.3: Flow chart of the proposed GPU implementation of OMP.

copied from the CPU memory to the GPU global memory. The **"cublasDgemv"** function from the cuBLAS API is used to perform the correlation operation with the residual (r_j): $\mathbf{D}_c{}^\dagger r_j$. Next, **"cublasIdamax"** is used to find the index with the largest projection, i.e. $k_j = \arg\max_j |\mathbf{D_c}{}^\dagger r_j|$. Finally, computing the least-square solution: $\boldsymbol{\alpha}_j = \arg\max_{\boldsymbol{\alpha}_j} \left\|\mathbf{y}_i - \mathbf{D}_{cj}^\dagger \boldsymbol{\alpha}_j\right\|_2^2$ using the API **"cublasDnrm2"** and then updating the provisional solution, $\mathbf{D}_c^\dagger \alpha_j$ using the **"cublasDaxpy'** are done. Once the OMP execution in GPU is over, the sparse coefficients are copied from the GPU global memory back to the CPU memory. A flowchart of the GPU implementation of the OMP algorithm is shown in Fig. 3.3.

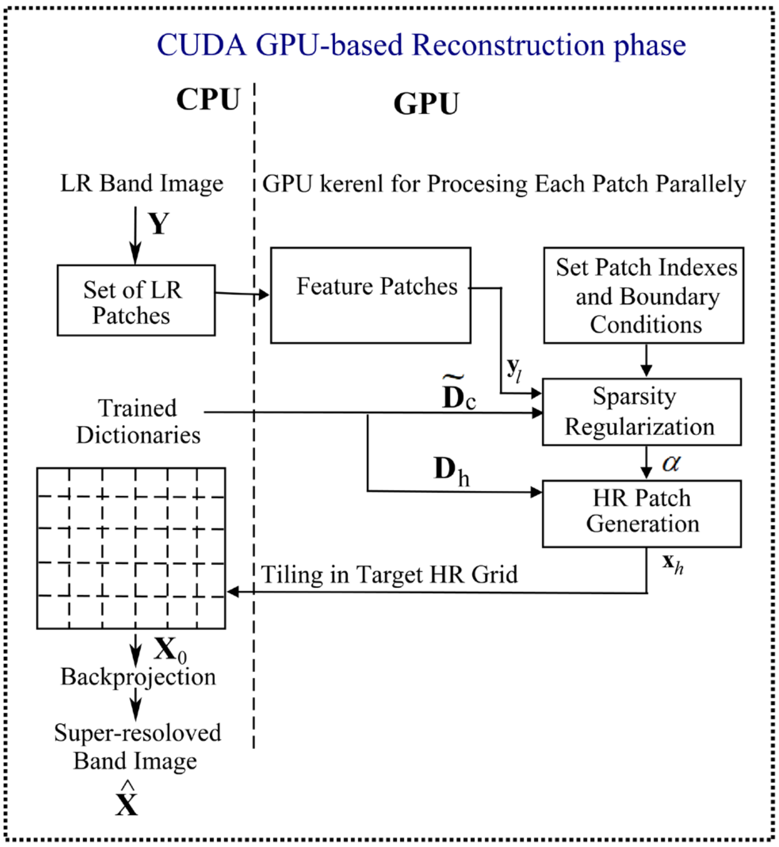


Figure 3.4: Schematic of proposed GPGPU SR image reconstruction method.

3.3.2.2 GPGPU-based super-resolution reconstruction of the MS image

Since image patches can be processed independently, this may be a key feature of the image data to explore using the GPU in the CUDA programming model. The proposed SR image reconstruction method is described in the second stage of Fig. 3.1 (blocks in green indicated). The parallel SR reconstruction flow is graphically shown in Fig. 3.4. A kernel function is defined by using *_global_* specifier on the global memory to perform the SR operation on each and every patches independently. This offers fine-grained thread-level parallelism of GPU; each thread is qualified for processing a patch. Three-level thread hierarchy is used: threads, thread block, and grids of blocks for the implementation. A grid of 65536 blocks, each having 1024 threads, is used for parallel patch processing. The input patches and the HR/LR dictionaries ($\mathbf{D}_h$ and $\mathbf{D}_\ell$) are transferred from the host (CPU) memory to the global memory (GPU). Next, sparse coefficients for each LR patch are obtained using the ℓ_1 feature-sign approach that runs on an individual thread. Once all the HR patches are reconstructed, they are transferred from the GPU back to the host CPU.

3.4 Results and Discussion

3.4.1 Dataset

RS multispectral images of Indian satellite LISS-IV have been procured from the NRSC data center (https://uops.nrsc.gov.in/). The images are in three bands (**band1:** green (0.52-0.59 μm); **band 2:** red (0.62-0.68 μm); **band 3:** NIR (0.77-0.86 μm)) with a spatial resolution of 5.8 meters. They were captured by the satellite during January 6, 2013 to April 10, 2018 over many places in India.

3.4.2 Experimental set-up

Experiments are performed on a Linux server having Intel® xeon® CPU running with Ubuntu 16.04 OS and equipped with 128 GB RAM and NVIDIA Tesla P100 GP-GPU card. Open source computer vision library (OpenCV) packages version of 3.3.1 and CUDA toolkit 9.0 are used for developing the software. Simulations are carried out using C++ and CUDA programming. The experimental settings used for the proposed method is shown in Table 3.1.

Table 3.1: Parameters setting

Dictionary Size	512
Patch size	7×7
No. of pixels in overlap for patch extraction patches	6
Regularization parameter (λ)	0.15

3.4.3 Performance evaluation

Reconstructed images using various methods for the same test image taken from LISS-IV are shown in Figs. 3.5 and 3.6 for different zooming factors. A small area

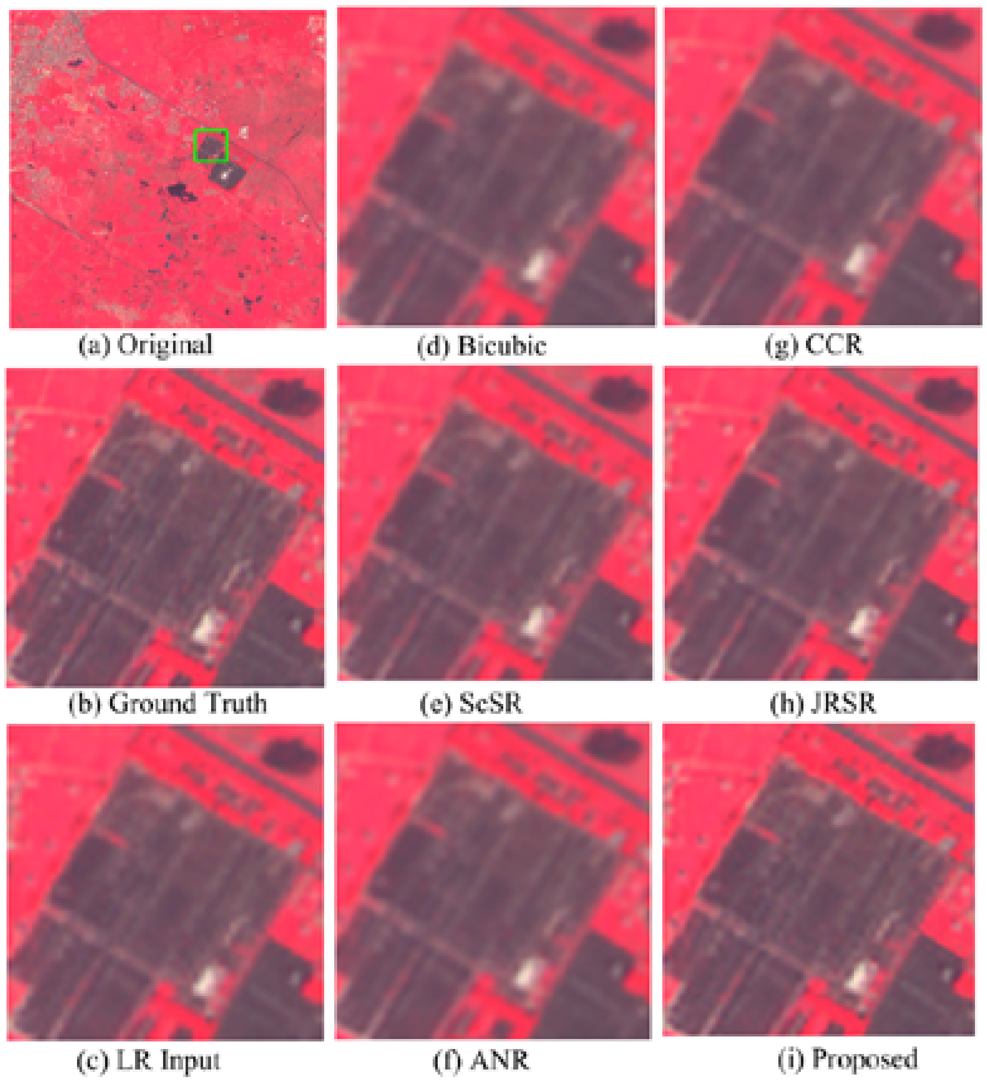


Figure 3.5: Visual outputs of different SR methods on the LISS-IV test image for upscaling factors 2 and 4. (Zoom in for better view)

marked in the original image by a green box is zoomed in and shown for different methods for visual analysis. Results indicate that the proposed method not only achieves lower reconstruction error, but also retains more structural features than others except JRSR for 4× zooming, where it shows comparable results. It is because most of the state-of-the-art methods are based on global dictionary learning; their visual results seem appropriate, but the details restored from the training may be poor. Furthermore, they have high computational cost and memory consumption. In MS remote sensing, global dictionary learning-based SR methods are not convenient for real time processing, because it is time consuming and expensive to produce the HR training data. On the contrary, the proposed method is based on adaptive

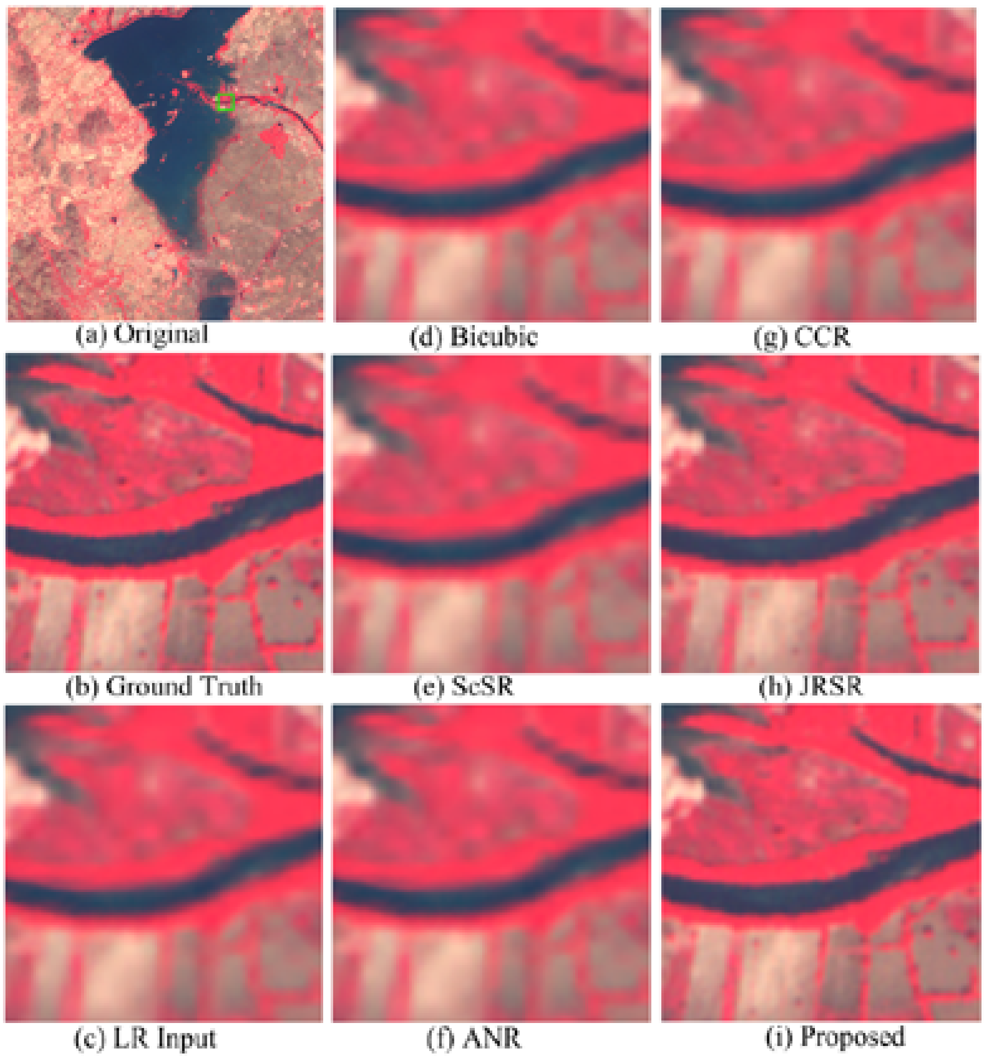


Figure 3.6: Visual outputs of different SR methods on the LISS-IV test image for upscaling factors 2 and 4. (Zoom in for better view)

dictionary learning, where only the test LR image is used for the dictionary training and SR reconstruction.

3.4.3.1 CPU implementation

To evaluate the efficacy of the proposed SR methods, a series of test images of sizes 128×128 to 3000×1500 are cropped from a real MS image of size 18133×16000; applied them band-wise as inputs to the proposed algorithm. The real RS multi-spectral image is only a representative image randomly selected from the LISS-IV dataset for simulation. For patch-based processing, we have extracted patches of size 7×7 from the cropped images. A coupled dictionary consisting of $\mathbf{D}_h$ of size 49×512, and $\mathbf{D}_\ell$ of size 196×512 is learned from the 10,0000 selected patch-pairs from the test LR multispectral image bands. Fig. 3.7 shows the learned LR and HR dictionaries obtained by the proposed feature extraction-based adaptive dictionary learning technique using KSVD.

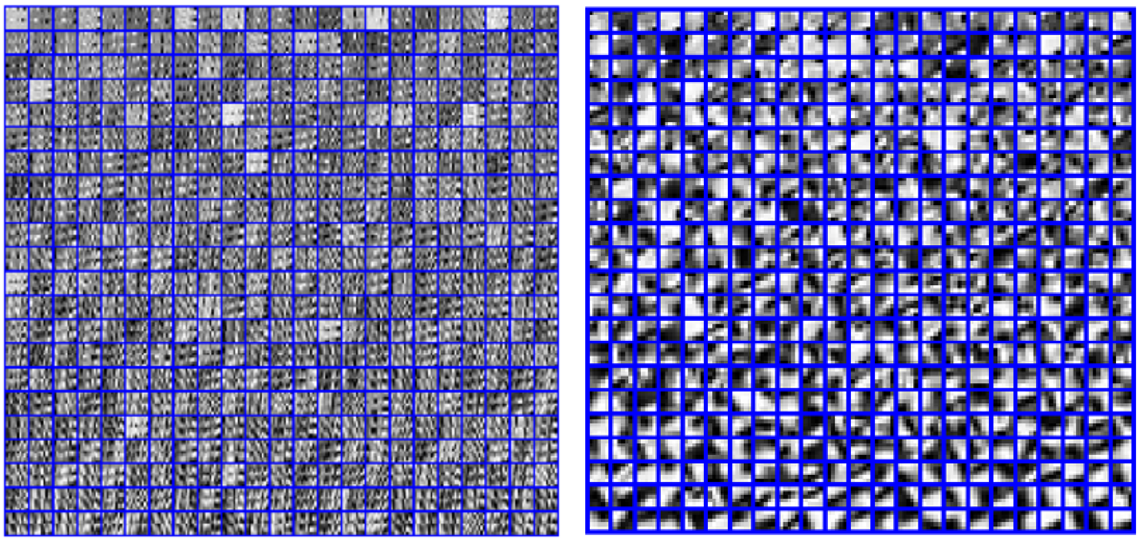

(a) LR dictionary **(b)** HR dictionary

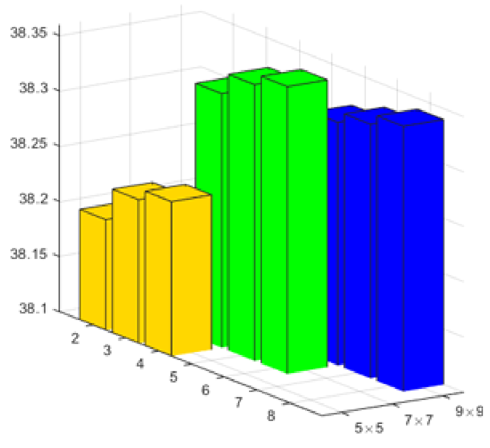
Figure 3.7: Learned LR and HR dictionaries using the proposed feature extraction-based KSVD algorithm.

Evaluations in terms of PSNR and SSIM measures, ERGAS [105], SAM [121], Q-index [113] and sCC [137] are done to validate the proposed method. Results are compared with four state-of-the-art learning-based methods: sparse coding super-resolution (ScSR) [118], anchored neighborhood regression (ANR) [98], clustering and collaborative representation (CCR) [131] and joint regularization-based SR (JRSR) [14] and two deep learning-based SR methods i.e. SRCNN [22] and VDSR [45]. Performance of the proposed method for 2x and 4x zooming are shown in Table 4.2 for a test image of size 512×512. It can be found that the proposed method clearly shows better results than others for 2× zooming. On an average, PSNR increases approximately by around 4.23 dB and 1.37 dB over bicubic for 2× and 4× zooming, respectively. The PSNR gain of the proposed method over the second best method i.e. JRSR is 0.67 dB for 2× zooming. But for 4x zooming, the proposed method is similar to the JRSR. It is found that the proposed method gives better performances over the two deep learning-based SR methods considered here as the training LISS-IV MS image database is relatively small ($\sim$ 200 images) for learning their models.

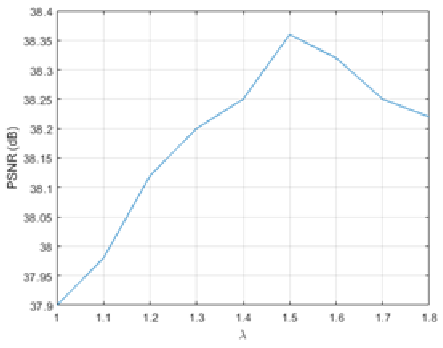
3.4.3.2 Parameter sensitivity analysis

While doing a parameter sensitivity study, the PSNR values obtained by the proposed methods are observed by changing only one parameter, while keeping the other parameters fixed. The efficiency of the proposed method is significantly affected by the patch size selection and the number of overlapping pixels (stride). Results on PSNR versus patch size as well as PSNR versus λ on the LISS-IV Test image for zooming factor 2 are shown in Fig. 3.8a. The results indicate that the PSNR of the proposed method is maximum when the patch size and overlapping pixel numbers are considered to be 7×7 and 6, respectively. Therefore, the proposed method experimentally fixes these parameters.

Changing the λ value from 1 to 1.8 gives different PSNR values, as shown in Fig. 3.8b for the LISS-IV Test image at zooming factor 2. It shows that the maximum PSNR value occurs at 1.5.



(a) PSNR versus the patch size and the number of overlapping pixels



(b) PSNR versus λ

Figure 3.8: PSNR plot vs. parameters.

Table 3.2: Performance Evaluation of Test Image size of 512 using different methods for upscaling factors 2 and 4

		×2								×4							
Metric	Band	**Bicubic**	**ScSR**	**ANR**	**CCR**	**JRSR**	**SRCNN**	**VDSR**	**Proposed**	**Bicubic**	**ScSR**	**ANR**	**CCR**	**JRSR**	**SRCNN**	**VDSR**	**Proposed**
PSNR (dB)	Band 1	38.36	40.23	40.02	40.02	41.78	39.46	40.18	43.22	32.10	32.65	33.18	33.30	33.66	32.23	32.39	33.55
	Band 2	34.64	37.02	35.92	36.03	37.98	35.76	36.27	39.25	27.56	28.23	28.47	28.60	28.91	27.93	28.08	28.85
	Band 3	31.72	33.47	33.13	33.34	35.62	32.62	33.24	34.92	27.86	28.48	29.02	29.37	29.56	28.05	28.25	29.45
	Average	**34.90**	**36.90**	**36.36**	**36.46**	**38.46**	**35.94**	**36.56**	**39.13**	**29.17**	**29.79**	**30.22**	**30.22**	**30.22**	**29.40**	**29.57**	**30.54**
SSIM	Band 1	0.949	0.966	0.961	0.959	0.967	0.950	0.961	0.981	0.819	0.838	0.842	0.845	0.866	0.813	0.812	0.855
	Band 2	0.922	0.952	0.940	0.938	0.953	0.938	0.943	0.972	0.725	0.757	0.760	0.765	0.800	0.730	0.735	0.780
	Band 3	0.900	0.938	0.919	0.920	0.947	0.917	0.921	0.950	0.724	0.749	0.753	0.762	0.786	0.735	0.745	0.771
	Average	**0.923**	**0.952**	**0.940**	**0.939**	**0.967**	**0.935**	**0.941**	**0.963**	**0.756**	**0.781**	**0.785**	**0.790**	**0.817**	**0.759**	**0.764**	**0.802**
ERGAS	Band 1	5.91	4.76	4.86	4.88	3.98	5.03	5.17	3.38	10.52	9.89	9.30	9.18	8.81	10.37	10.18	8.96
	Band 2	12.00	9.12	10.36	10.21	8.16	10.12	9.9	7.06	17.59	16.27	15.84	15.59	15.04	17.23	16.54	15.24
	Band 3	5.54	4.54	4.72	4.60	3.54	4.82	4.6	3.81	11.48	10.68	10.04	9.66	9.43	11.21	11.32	9.59
	Average	**7.82**	**6.14**	**6.65**	**6.56**	**5.22**	**6.65**	**6.55**	**4.75**	**13.20**	**12.28**	**11.720**	**11.48**	**11.09**	**12.93**	**12.68**	**11.26**
SAM	Band 1	0.029	0.023	0.024	0.024	0.019	0.023	0.024	0.012	0.052	0.049	0.046	0.045	0.044	0.0494	0.050	0.042
	Band 2	0.058	0.044	0.050	0.049	0.040	0.052	0.047	0.031	0.084	0.078	0.076	0.074	0.071	0.080	0.078	0.071
	Band 3	0.027	0.022	0.023	0.022	0.018	0.024	0.022	0.016	0.054	0.046	0.047	0.045	0.043	0.052	0.049	0.042
	Average	**0.038**	**0.030**	**0.032**	**0.032**	**0.026**	**0.033**	**0.031**	**0.019**	**0.0633**	**0.0577**	**0.0563**	**0.0547**	**0.0527**	**0.060**	**0.059**	**0.0516**
UIQI	Band 1	0.777	0.834	0.829	0.808	0.842	0.801	0.831	0.897	0.450	0.507	0.518	0.519	0.596	0.470	0.492	0.551
	Band 2	0.816	0.830	0.859	0.846	0.878	0.821	0.865	0.929	0.457	0.520	0.526	0.530	0.607	0.472	0.503	0.560
	Band 3	0.769	0.793	0.784	0.798	0.832	0.775	0.790	0.792	0.412	0.467	0.472	0.480	0.549	0.425	0.441	0.505
	Average	**0.788**	**0.819**	**0.824**	**0.817**	**0.851**	**0.799**	**0.828**	**0.872**	**0.440**	**0.498**	**0.505**	**0.509**	**0.584**	**0.455**	**0.478**	**0.538**
sCC	Band 1	0.794	0.835	0.853	0.842	0.835	0.805	0.821	0.885	0.406	0.427	0.467	0.476	0.513	0.412	0.421	0.484
	Band 2	0.843	0.890	0.885	0.851	0.876	0.861	0.873	0.923	0.426	0.450	0.480	0.492	0.529	0.432	0.438	0.503
	Band 3	0.851	0.894	0.877	0.883	0.898	0.876	0.882	0.865	0.401	0.429	0.456	0.480	0.503	0.413	0.419	0.481
	Average	**0.830**	**0.873**	**0.871**	**0.859**	**0.870**	**0.847**	**0.858**	**0.891**	**0.411**	**0.435**	**0.468**	**0.483**	**0.515**	**0.419**	**0.426**	**0.489**

3.4.3.3 Convergence test and Complexity analysis

An empirical study is conducted to evaluate a proposed method using KSVD, where convergence is assessed by determining the PSNR of the reconstructed image. Fig. 3.9 shows the convergence plot for band 1 at 2× zooming, with 30 iterations considered. The highest PSNR is observed at the 20th iteration, marking it as the optimal stopping point for the algorithm. This peak suggests that PSNR increases gradually up to the 20th iteration, after which it remains nearly constant, indicating a stabilization in PSNR gains.

The complexity of the proposed method is influenced by various computational processes. The most significant computational loads come from the K-SVD dictionary learning and optimization steps. Specifically, the K-SVD step (step 5 of Algorithm 2) iteratively updates dictionary atoms and performs sparse coding of all patches, has a complexity of $\mathcal{O}(T \times (n \times t \times k \times r^2 + k \times (m \times r^2 + r^3)))$. Here, T is the number of iterations, n is the number of patches, t is the sparsity level, k is the number of dictionary atoms, r is the patch size, m is the number of feature maps. Next, major computation comes from the patch optimization (step 7 of Algorithm 2) with $\mathcal{O}(n \times k \times r^2)$. When considering CPU execution, the overall complexity of the proposed method can be expressed as $\mathcal{O}((T \times (n \times t \times k \times r^2 + k \times (m \times r^2 + r^3))) + n \times k \times r^2)$. Assuming effective utilization of GPU capabilities for parallel processing of patches, the total time complexity reduces to $\mathcal{O}((T \times (t \times k \times r^2 + k \times (m \times r^2 + r^3))) + k \times r^2)$.

Table 3.3: CPU vs GPU speed-up for different dictionary size.

Dictionary Size	**CPU(secs)**	**GPU (secs)**	**Speed-up**
256	105	10	10.5
512	193	17	11.3
1024	363	27	13.44

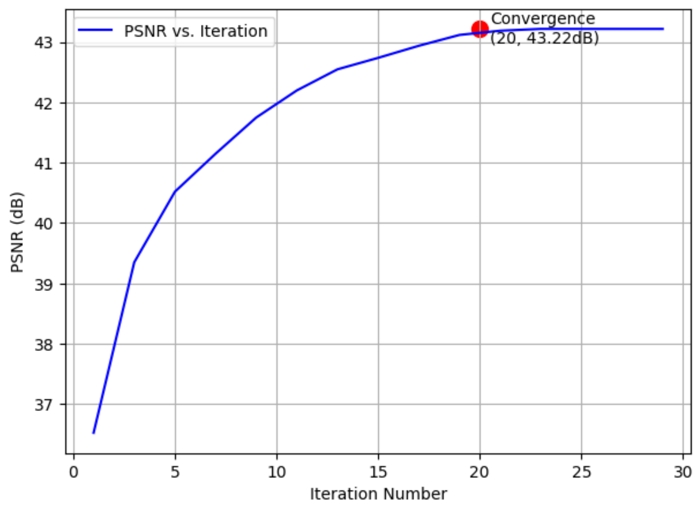


Figure 3.9: Convergence analysis of the proposed method for 2× zooming: PSNR vs number of iterations.

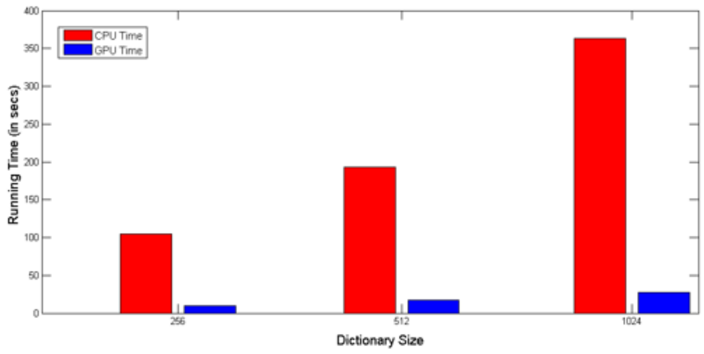


Figure 3.10: Execution time for CPU and GPU implementations with varying dictionary size.

3.4.3.4 *CUDA-based GPGPU implementation*

Experiments are carried out to analyze the scalability of our implementation in terms of dictionary and input image sizes. Table 3.3 shows the execution time taken for CPU- and GPU-based implementations for different dictionary sizes. Different dictionaries are trained using the test LR image of size 512 × 512. The $\mathbf{D}_h$ and $\mathbf{D}_\ell$ are trained using a fixed number of sample patches of around 100000. As the sample patches are kept fixed, it takes the same execution time for different upscaling factors on the CPU. As the dictionary size increases, the speed of GPU implementation increases from 4 to 5 times as shown in Fig. 3.10. In the reconstruction, the image size (number of pixels) is varied from 128×128 to 1024×1024. Table 3.4 compares the execution time for CPU and GPU-based reconstructions for 2x and 4x zooming.

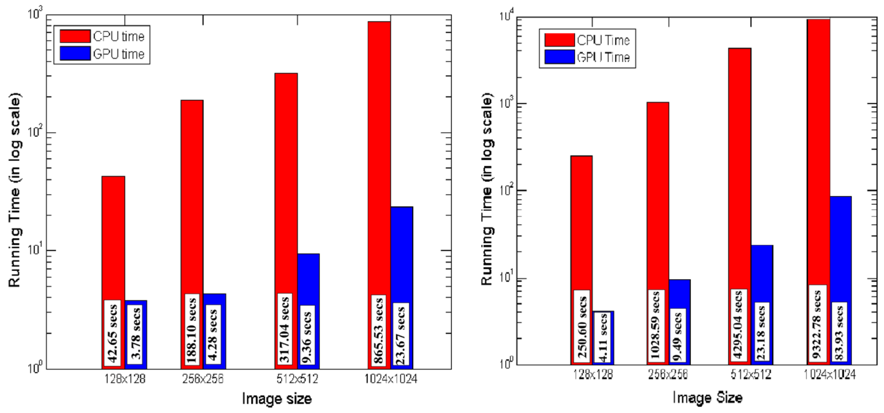


Figure 3.11: Reconstruction time of CPU and GPU implementation for different image sizes with zooming factor 2.

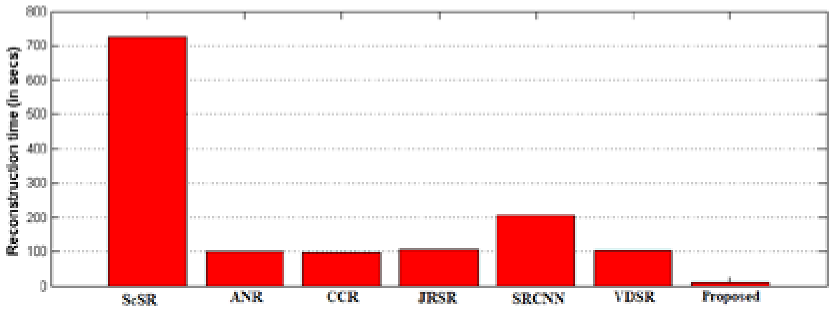


Figure 3.12: Reconstruction runtime comparison of different methods for zooming factor 2.

As the image sizes increases, the GPU speed-up improves steadily from 11 to 36 as compared to the CPU for 2x zooming. On the other hand, the proposed method takes about 4-83 secs. and reduces the reconstruction time by 60 to 111 times as compared to their CPU counterparts. Fig. 3.11 shows the graphical representations of reconstruction times of CPU and GPU-based implementations for the proposed method for 2x and 4x zooming.

The proposed method with GPU-CUDA implementation and the existing methods are compared in terms of reconstruction time in Fig. 3.12 for a test band image of size 512×512. When compared to the existing methods for 2× zooming factor, the GPU-CUDA based proposed method yields faster results. Since the spatial size of real RS images is significantly large, it is very important to process these images within a real-time. Our efficient CUDA-GPU based SR algorithm can easily process larger RS images (up to 3000 × 1500 image size) within a few minutes for different upscaling factors. Execution times taken for dictionary training and reconstruction

Table 3.4: CPU vs GPU Reconstruction time for zooming factors 2 and 4.

Image size		×2					×4				
		Proposed method		**Average CPU time (secs) (ts)**	**Average GPU time (secs) (tp)**	**Speed up (tp)**	**Proposed method**		**Average CPU time (secs) (ts)**	**Average GPU time (secs) (tp)**	**Speed up (tp)**
		CPU time (secs)	**GPU time (secs)**				**CPU time (secs)**	**GPU time (secs)**			
128 × 128	**Band 1**	43.99	3.80	42.75	3.78	11.28	249.01	4.11	250.60	4.11	60.97
	Band 2	43.99	3.80				252.35	4.12			
	Band 3	43.68	3.79				250.44	4.11			
256 × 256	**Band 1**	188.10	4.25	188.10	4.28	29.93	1028.22	9.54	1028.59	9.49	108.38
	Band 2	188.10	4.27				1029.03	9.42			
	Band 3	187.11	4.32				1028.54	9.53			
512 × 512	**Band 1**	318.20	10.02	317.04	9.36	33.87	4295.23	23.01	4295.04	23.18	186.73
	Band 2	315.59	10.02				4294.02	23.42			
	Band 3	317.34	8.05				4295.44	23.13			
1024 × 1024	**Band 1**	856.20	24.01	865.53	23.67	36.56	9322.17	83.02	9322.78	83.93	111.07
	Band 2	907.20	24.01				9324.17	82.55			
	Band 3	834.20	23.00				9322.01	83.22			

Table 3.5: GPU dictionary training/reconstruction time with upscale factor 2 and 4.

Image size		**x2**		**x4**	
		Dictionary training time (secs)	**Reconstruction time (secs)**	**Dictionary training time (secs)**	**Reconstruction time (secs)**
2048 × 2048	**Band 1**	49.61	83	53.21	403
	Band 2	49.41	83	53.25	402
	Band 3	50.42	82	49.68	403
3000 × 1500	**Band 1**	49.01	147	54.00	415
	Band 2	53.12	146	56.20	415
	Band 3	54.02	147	56.02	415

for different larger MS remote sensing images using the proposed CUDA-GPU-based parallel SR algorithm are shown in Table 3.5 for upscale factors 2 and 4. The sequential counterpart in CPU cannot process real RS images due to their memory limitation.

3.4.4 Application: Land cover classification

The classification studies have been performed on SR images obtained by using different techniques to interpret various regions present in the MS remote sensing image. Envi classic 5.1 is used for classification and analysing the classified results. The unsupervised classification has been performed using k-means clustering on the test image for different methods. Five classes are defined on the test image using the Envi classification tool: vegetation (green), barren soil (navy blue), water (yellow), road and building (red), and dense Building (sky blue). The results indicate that the proposed method efficiently separates the classes that are quite similar to that of the ground truth. But, in case of the LR image and results of other SR methods, some

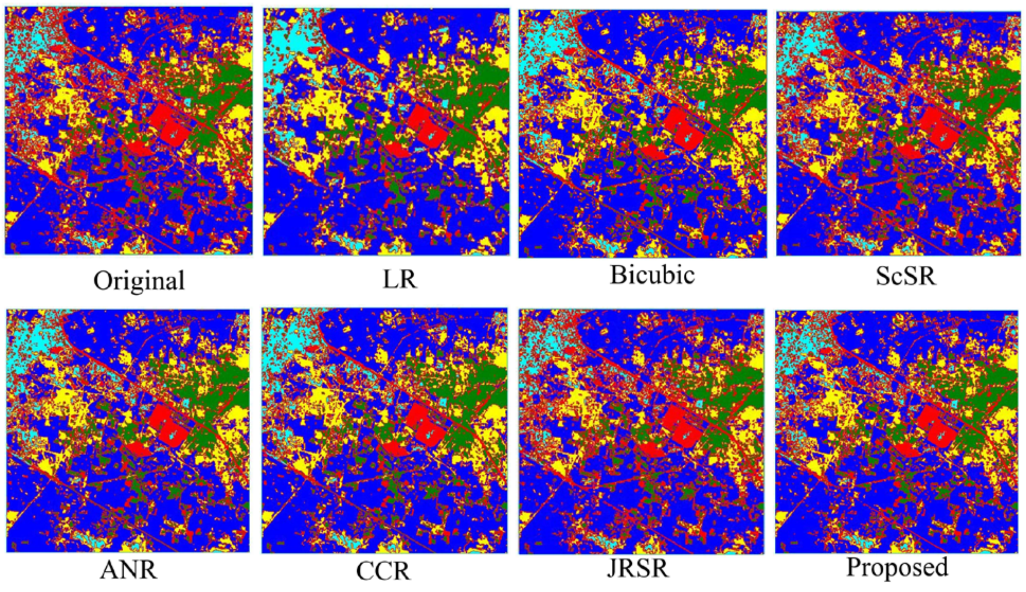


Figure 3.13: Reconstruction time for CPU and GPU implementations for different image sizes with zooming factors 2 and 4.

of the classes are not clearly identified and some regions are misclassified. Results of unsupervised classification on test images for different methods are shown in Fig. 5.14. The percentage of classification accuracy on each class of the test image for different methods is given in Table 3.6. The classification accuracies on each class of the proposed method are closed to that of the ground truth. While the class-wise accuracy rates of the ScSR, ANR and CCR deviate from the ground truth. Since the image quality of the proposed method is comparatively higher than other methods, the classification error between the proposed method and the ground truth image is low, when compared to other methods. Therefore, it can be concluded that the reconstruction accuracy of the proposed method is clearly superior to its counterparts.

Table 3.6: Classification accuracy on each class of Test Image using unsupervised classification for different methods.

Class	**Ground truth** (%)	**LR Input** (%)	**Bicubic** (%)	**ScSR** (%)	**ANR** (%)	**CCR** (%)	**JRSR** (%)	**Proposed** (%)
Dense Building (Sky blue)	3.60	3.16	2.98	3.16	3.02	3.04	3.45	3.40
water (yellow)	17.63	16.67	17.94	17.92	17.81	17.75	17.23	17.33
Barren soil (Navy blue)	51.75	53.01	50.13	50.51	50.45	50.54	51.68	51.69
Road and Building (Red)	20.09	19.69	21.25	20.94	21.17	21.14	20.65	20.58
Vegetation (Green)	6.90	7.45	7.45	7.45	7.53	7.52	6.98	6.90

3.5 Conclusion

A GPU-CUDA accelerated SISR framework for MS remote sensing images using adaptive dictionary learning and sparse representations is developed in this chapter. We have found that adaptive dictionary learning is highly preferred for real-time RS applications. A new feature extraction step is also introduced for restoring the reconstructed image with sharp edges and high-frequency details. Experimental results show that the proposed method offers advantages in terms of better visual outputs and objective criteria compared to the state-of-the-art. Highly parallelized algorithms based on CUDA-GPU are designed for the proposed method, which can provide improvements in computational time by manifolds than their CPU-based sequential counterparts. It can be concluded that the proposed framework would be highly useful for near real-time applications. However, the proposed method may not adequately preserve the sharpness and clarity of edges and other high frequencies details as accurately as needed for RS applications, since it is very difficult to learn the statistical properties of edges from a single image.

Since, edge preservation is necessary in RS super-resolution for applications, like object detection and surveillance because these tasks often require precise and accurate identification of objects and their boundaries. In the future, we will propose a more effective SISR framework that uses a global dictionary training to learn more edge-based image properties. In order to reconstruct edge-enhanced SR images, a joint regularization problem can be used by incorporating edge-oriented priori information. It is also essential to develop highly parallelized CUDA-based algorithms for KSVD-based dictionary learning and edge-enhanced reconstruction phases, which can further speed-up the complete MS super-resolution process.