

CHAPTER 4

Development of Edge Preserving Remote Sensing Single Image Super-resolution based on Global Dictionary Learning and Sparse Representations

4.1 Introduction

Preservation of textural or structural features, particularly the edges, is crucial in the SR of RS images. These features may be used as a prior information in dictionary learning to obtain edge-enhanced SR, as learning a representative dictionary in terms of edges is a key challenge in sparse representation models. Central to this process is the use of an overcomplete dictionary, which can be constructed from two orthobasis taken from well-known transforms, like wavelets, Fourier transforms, etc. Alternatively, it can be constructed through optimization to enhance sparsity in signal representation. Learned dictionaries show superior performance compared to the pre-defined transforms in terms of sparsity solutions because they incorporate prototype signal-atoms, taken from the given image itself, allowing signals to be represented by sparse linear combinations of these atoms.

The challenge of constructing a representative dictionary that effectively captures edge details is significant in sparse representation models. Although In the previous chapter, the previous chapter explores adaptive dictionary learning to reconstruct SR remote sensing images, but it is quite challenging to learn the edge-based properties from a single image. In order to obtain edge-enhanced SR, the global overcomplete dictionary learning method is very much convenient as it can afford a large suitable training set and fully utilizes image priors to efficiently improve edges and textural features. An overcomplete dictionary is used for representing signals sparsely, can be

Regularization plays an important role in sparse representation-based SR because it is used to impose constraints on the solution of the sparse representation problem to prevent overfitting. One common form of regularization used in SR is known as “edge-preserving regularization.” This form of regularization is used to preserve the edges and other important features in the reconstructed HR image. While the standard regularization methods can result in overly smooth images that lack fine details and sharp edges. These methods typically penalize the total variation of the reconstructed image, which can result in a solution that is too smooth. Edge-preserving regularization, on the other hand, penalizes changes in intensity across edges, while allowing for more variation in smoother regions of the image. This results in an important tool in achieving high-quality SR. To restore both sharp edges and smooth structures simultaneously, a joint regularization technique may also be used by combining two regularization terms such as the non-local total variation (NLTV) regularization term that encourages smoothness in the output image and an edge-enhancing-based regularization term that penalizes deviations from the edges in the original LR image.

The computational complexity of such algorithms is generally very high. Therefore, CUDA-enabled GPU paradigm is the best choice for handling the computational costs of such SISR algorithms, depending on the algorithmic complexity and dimensions of the real RS images.

In this chapter, a coupled dictionary based on scale-invariant feature transform (SIFT) keypoints as well as non-keypoints image patches are learned that effectively preserve edges of an image. In particular, we propose a joint sparse reconstruction model based on SIFT-guided and NLTV regularization priors to preserve high frequency information as well as smooth structures of the reconstructed image. CUDA-based implementation of the alternating direction method of multipliers (ADMM) technique is also carried out to solve the above joint problem and reconstruct a real RS image within a reasonable time. The proposed method not only retains edges, but also restores visually enhanced images for various zooming factors.

4.1.1 Main contributions of the chapter

The major contributions of the works done in this chapter may be summarized as follows:

- (i) Proposed keypoints and non-keypoints features-based novel overcomplete dictionary learning techniques for RS image SR. An external MS remote sensing database is used to learn multiple coupled dictionaries- SIFT-based keypoints and non-keypoints patch-based dictionary pairs to preserve structural and textural features effectively. Compared to the traditional patch-based dictionary alone, the proposed dictionaries can preserve the high frequency information precisely.
- (ii) Proposed a joint sparse reconstruction model by combining SIFT-driven keypoints and NLTV regularization priors. Different sub-problems are solved iteratively using the CUDA-based ADMM.
- (iii) Extensive simulations are demonstrated on two publicly available RGB RS and two real MS remote sensing datasets for various scaling factors to show that the proposed model outperforms state-of-the-art techniques both visually and quantitatively. The proposed parallel framework for SR shows good potential to process MS remote sensing images up to 2048×2048 in a few minutes for $4\times$ upscaling.

The rest of the chapter is organized as follows: Section 4.2 presents some prior art on SIFT and NLTV regularization. Section 4.3 discusses about the proposed methodology and joint sparse reconstruction model, including its implementation on CUDA-GPU platform. Experimental results using different datasets are discussed in Section 4.4. Finally, Section 4.5 concludes the chapter.

4.2 Prior Art

4.2.1 Scale-invariant feature transform (SIFT)

SIFT is an algorithm in computer vision utilized to detect and describe local features present within an image. SIFT can detect features that are invariant to changes in scale, rotation, and illumination. This makes it a powerful tool for object recognition, image stitching, and 3D reconstruction. The SIFT algorithm involves several steps. The first step is to construct a scale-space representation of the image. This involves creating a series of blurred images at different scales, using a Gaussian kernel with increasing standard deviation. The purpose of this is to detect features at different scales or multiresolution levels.

The next step is to identify local extrema in the scale-space representation. These extrema correspond to features that are invariant to scale changes. The algorithm searches for extrema across different scales and space, using a difference-of-Gaussians (DoG) function. Once the extrema are detected, the algorithm performs keypoints localization to improve their accuracy. This involves fitting a quadratic function to the scale-space representation at each keypoint and discarding keypoints with low contrast or those that are located on edges. The orientation of each keypoint is then assigned by computing the dominant orientation of gradient orientations within a circular region around the keypoint. This step ensures that the descriptors are invariant to rotation.

The keypoint descriptor is generated for each keypoint by computing a histogram of orientations within a spatial region by considering a 16×16 window around the keypoint. This window is further divided into 16 sub-blocks, each of size 4×4 . Gradients are calculated for each pixel within each sub-block. An orientation histogram, usually comprised of 8 bins spanning 360 degrees, is constructed using these gradients. The distribution of gradient orientations within the sub-block is represented by the orientation histogram. As a result, the final size of keypoint descriptor is

determined by multiplying the number of sub-blocks (16) by the number of bins per histogram (8), obtaining a descriptor size of 128 elements. The resulting descriptor is a vector of features that describes the local image content around the keypoint. Overall, SIFT is a powerful algorithm for detecting and describing local features in images. It has been extensively used in a variety of applications and has paved the way for many subsequent feature detection and matching techniques.

4.2.2 Non-local total variation (NLTV) regularization

NLTV regularization is a technique used in image processing and computer vision to smooth out noisy images, while preserving the sharp edges and details of the image. Unlike traditional total variation regularization, NLTV takes into account the local statistics not only in the pixel neighborhood but also searches for similar patches within the image and perform the required filtering operation. The NLTV-based image denoising model is composed of up of two terms: NLTV and data-fidelity, which can be expressed as [63]:

$$\tilde{\mathbf{x}} = \arg \min_{\mathbf{x}} J_{NLTV}(\mathbf{y}) + \frac{\lambda}{2} \|\mathbf{y} - \mathbf{x}\|^2, \quad (4.1)$$

J_{NLTV} is the NLTV regularization term, which can be expressed as follows:

$$J_{NLTV} = \sum_{i \in \Omega} \sqrt{\sum_{j \in M_i} (\mathbf{y}_i - \mathbf{x}_j)^2 \mathbf{w}(i, j)}, \quad (4.2)$$

where the denoised, original, and noisy image patches are represented by $\tilde{\mathbf{x}}$, \mathbf{x} , and \mathbf{y} , respectively. M_i refers to a square neighborhood of size $N \times N$ centered at pixel i . λ is the regularization parameter that controls the amount of smoothing applied to the image, and $w(i, j)$ is the non-local weight function that measures the similarity between the image patches centered at i and j in image \mathbf{x} . The non-local weight function $\mathbf{w}(i, j)$ is defined as follows:

$$w(i, j) = \exp \left(-\frac{\|P_i(\mathbf{Y}) - P_j(\mathbf{Y})\|_{\sigma^*}^2}{2h^2} \right), \quad (4.3)$$

where $P_i(\mathbf{Y})$ and $P_j(\mathbf{Y})$ denote the image patches of $n \times n$ size centered at i and j , $\|\cdot\|_{\sigma^*}^2$ utilizes a Gaussian kernel to assign weights to the Euclidean norm, and h is a parameter of filtering, depends on the noise level. The non-local weight function captures the degree of similarity between the image patches across the entire image, which allows NLTV regularization to preserve edges and details that could be lost with traditional local regularization techniques.

4.3 Proposed method

The proposed method consists of two phases: A) development of a novel SR approach using sparse coding and dictionary learning, and B) design of massively parallel algorithms on GP-GPU using off-the-shelf and user defined CUDA kernels. First, it is implemented sequentially; the coupled dictionaries are learnt from an external MS image dataset and the target SR image is reconstructed using a joint sparse reconstruction model. Next, in order to exploit the data level parallelism and increase the computational efficiency, massively parallel algorithms are designed, especially for the computationally exhaustive components. In the following, we elaborate on each stage:

4.3.1 Feature extraction

It extracts high-frequency information from input LR images to improve sparse representation accuracy. In this chapter, to carry out the proposed feature extraction and dictionary training both on natural RGB images and MS images in a uniform framework, first, we have converted the real MS data into false RGB images. Next the false RGB images are converted into the YC_bC_r format on which the proposed

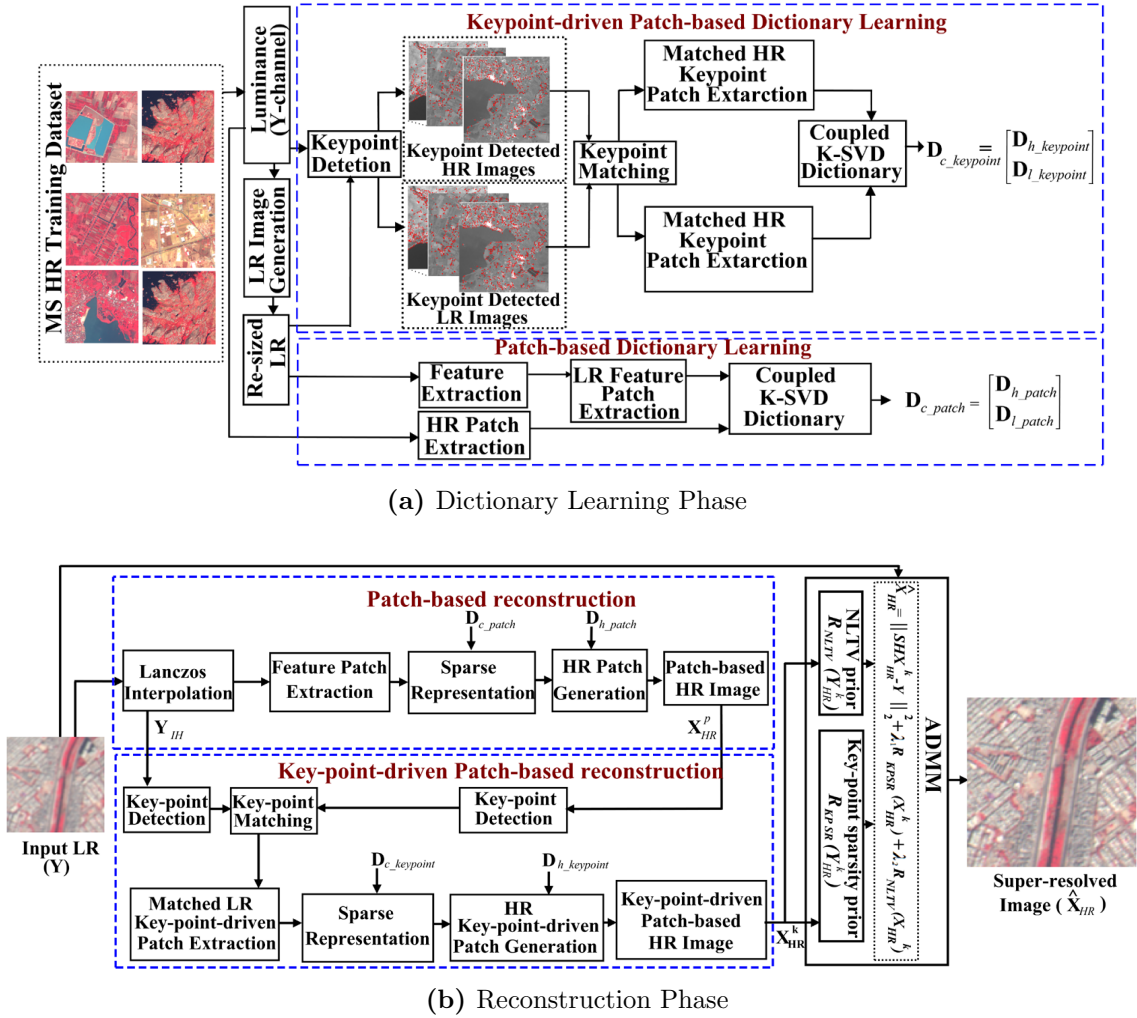


Figure 4.1: Schematic representation of the proposed edge enhanced framework for remote sensing SR.

dictionary learning and sparse reconstruction algorithms are carried out. The luminance channel Y is blurred and then downsampled in order to produce the LR training images. The reason for selecting only the luminance channel (Y) is due to the fact that human eyes are more sensitive towards the luminance information compared to the chrominance or colour information. Besides, the chrominance information of RS images are not considered as one of the key components in the dictionary learning method. The LR training images are then resized to generate upscaled LR images of the same size as the original HR images. Next, we apply the edge-enriched feature extraction method on them, which was already discussed in Chapter 3.

Table 4.1: Parameters used in SIFT keypoint detection and matching.

Number of keypoints	Keypoint scale	Sigma	Number of octaves	τ_k	Kernel size	τ_m
300~400K	3	1.6	4	0.007	15×15	1.5

4.3.2 Feature-enhanced patch-based multiple dictionary learning

4.3.2.1 Keypoints-based coupled dictionary learning

A schematic of the proposed MS image dictionary learning is shown Fig. 4.1a. In the dictionary learning stage, two sparse representations-based coupled dictionaries are learned using keypoint- and patch-based features, respectively. In particular, scale invariant feature transform (SIFT)-based keypoints-driven and non-keypoints patch-based multiple coupled dictionaries are learned. In the first case, the potential keypoints are detected from the HR and LR training images using the SIFT detector and then, matched keypoints between HR and re-sized LR training images are identified based on SIFT descriptors. The number of optimal keypoints are detected using the Taylor series expansion by rejecting those keypoints whose contrasts are less than a certain threshold (τ_k). Lowering the value of τ_k has no effect on the performance of the proposed technique, but it increases the number of unstable low contrast keypoints, which increases the computation time of the proposed method. On the other hand, it has a significant impact on the performance of the proposed method for higher value of τ_k since less possible keypoints are detected. The value of τ_k is selected such that the performance and computational time of the proposed method are not compromised. Keypoints are considered to be matched if Euclidean distance between the corresponding SIFT descriptors are lower than a given threshold (τ_m). The number of potential keypoints and parameters used for keypoints detection and matching using the SIFT are shown in Table 4.1. Next, local patch vectors surrounded by matched keypoints from both HR and re-sized LR training images, i.e, $\mathbf{X}_{keypoint} \in \mathbb{R}^{n_h \times K_p}$ and $\mathbf{Y}_{keypoint} \in \mathbb{R}^{m_l \times K_p}$, respectively are extracted for dictionary training. Here, K_p is the total numbers of matched patches. Using the combined HR-LR keypoint-driven

patch dataset $\mathbf{Y}_{c_keypoint} = [\mathbf{X}_{keypoint}; \mathbf{Y}_{keypoint}]$, the keypoint-driven HR-LR dictionary pairs $\mathbf{D}_{h_keypoint} \in \mathbb{R}^{m_h \times K}$ and $\mathbf{D}_{\ell_keypoint} \in \mathbb{R}^{m_\ell \times K}$ are jointly learned in the form of a coupled dictionary $\mathbf{D}_{c_keypoint} = [\mathbf{D}_{h_keypoint}; \mathbf{D}_{\ell_keypoint}]$, where m_ℓ and m_h denote the sizes of feature patches encircled by matched keypoints of LR and HR images, respectively (shown in the upper half of Fig. 4.1a). K is the dictionary size.

4.3.2.2 Non-keypoints-based coupled dictionary learning

In the traditional patch-based dictionary learning, the coupled dictionary \mathbf{D}_{c_patch} , consisting of patch-based HR-LR dictionary pairs $\mathbf{D}_{h_patch} \in \mathbb{R}^{n_h \times K}$ and $\mathbf{D}_{\ell_patch} \in \mathbb{R}^{n_\ell \times K}$ are jointly learned using the combined HR-LR patch dataset $\mathbf{Y}_{c_patch} = [\mathbf{X}_{patch}; \mathbf{Y}_{patch}]$. As explained in Section 3.3.1.2, Chapter 3, the HR patch matrix $\mathbf{X}_{patch} \in \mathbb{R}^{n_h \times P}$ is formed by directly extracting HR patch vectors containing most relevant information from the HR training images, while LR feature patch matrix $\mathbf{Y}_{patch} \in \mathbb{R}^{n_\ell \times P}$ is formed by stacking the LR feature patch vectors corresponding to each patch location; the LR feature vector is the concatenation of four filtered-image patch vectors for the selected patch location. The lengths of the HR patch vectors and the corresponding LR feature patch vectors are represented by the vectors n_h and n_ℓ , respectively and P is the total number of patch vectors extracted for the dictionary training. The above strategy of selection of feature patch vectors ensures that each LR patch incorporates its adjoining information, which is helpful for improving compatibility within neighboring patches in the super-resolved image. Both SIFT-based keypoints-driven and non-keypoints patch-based approaches consider the same patch size. Patch size selection is described in detail in Section 4.4.6.

Assuming that the two coupled dictionary pairs $\mathbf{D}_{c_keypoint} = [\mathbf{D}_{h_keypoint}; \mathbf{D}_{\ell_keypoint}]$, as well as $\mathbf{D}_{c_patch} = [\mathbf{D}_{h_patch}; \mathbf{D}_{\ell_patch}]$, share the common sparse representation matrices \mathbf{Z}_k and \mathbf{Z}_p , respectively, therefore, the coupled dictionaries $\mathbf{D}_{c_keypoint}$ and

\mathbf{D}_{c_patch} can be trained by solving the following minimization problems, respectively:

$$\begin{aligned} \min_{\{\mathbf{D}_{c_keypoint}, \mathbf{Z}_k\}} & \|\mathbf{Y}_{c_keypoint} - \mathbf{D}_{c_keypoint} \mathbf{Z}_k\|_F^2 + \lambda \|\mathbf{Z}_k\|_1 \\ \text{s.t.} & \|\mathbf{D}_{c_keypoint}(:, i)\|_2^2 \leq 1, \quad i = \{1, \dots, K\}. \end{aligned} \quad (4.4)$$

Similarly,

$$\begin{aligned} \min_{\{\mathbf{D}_{c_patch}, \mathbf{Z}_p\}} & \|\mathbf{Y}_{c_patch} - \mathbf{D}_{c_patch} \mathbf{Z}_p\|_F^2 + \lambda \|\mathbf{Z}_p\|_1 \\ \text{s.t.} & \|\mathbf{D}_{c_patch}(:, i)\|_2^2 \leq 1, \quad i = \{1, \dots, K\}, \end{aligned} \quad (4.5)$$

where K is the number of dictionary atoms or size of the dictionary, which is common for both the learned dictionaries, and λ used in Eqs. 4.4 and 4.5 represent the sparsity regularization parameter. Section 4.4.6 describes in detail how the value of λ is chosen. In order to solve Eqs. 4.4 and 4.5 approximately, the coupled K-SVD dictionary learning algorithm detailed in [80] is used.

4.3.3 SR Reconstruction

In the reconstruction phase, the super-resolved image ' $\hat{\mathbf{X}}_{HR}$ ' is obtained through two reconstruction stages in cascade; one based on the patch-based reconstruction and the other based on the keypoint-driven patch-based reconstruction. The proposed framework for reconstruction phase is shown in Fig. 4.1b. First, the input LR image ' \mathbf{Y} ' is upsampled to the required scale i.e. ' \mathbf{Y}_{IH} ' using Lanczos interpolation, and then passed through the feature extraction step as mentioned in section 4.3.1 to extract four high-frequency feature maps (the filtered images obtained by four high-pass filters: Sobel in x-, and y-directions, DoG, and Butterworth) of equal size. From each feature map, feature patches of size $\sqrt{n} \times \sqrt{n}$ (n : total number of pixels in a patch) with one pixel overlap are obtained. Eventually, the four feature patches from a given pixel location are vectorized and concatenated into a single feature vector $\mathbf{y}_{\ell_i}^p$ with a dimension of $4n \times 1$. Now, each LR patch $\mathbf{y}_{\ell_i}^p$ is sparsely represented as $\boldsymbol{\alpha}_i^p$ using the feature-sign method [118] from the corresponding feature vectors

using the already trained patch-based LR and HR dictionaries, i.e. by minimizing the following optimization problem:

$$\boldsymbol{\alpha}_i^p = \min_{\boldsymbol{\alpha}_i^p} \|\tilde{\mathbf{D}}_{c_patch} \boldsymbol{\alpha}_i^p - \tilde{\mathbf{y}}_i^p\|_2^2 + \lambda \|\boldsymbol{\alpha}_i^p\|_1, \quad (4.6)$$

where $\tilde{\mathbf{D}}_{c_patch} = \begin{bmatrix} \mathbf{D}_{\ell_patch} \\ S\mathbf{D}_{h_patch} \end{bmatrix}$, S represents the overlapped region of pixels between the target and the previously reconstructed HR image, and $\tilde{\mathbf{y}}_i^p = \begin{bmatrix} \mathbf{y}_{\ell_i}^p \\ \mathbf{g} \end{bmatrix}$, \mathbf{g} represents pixels of the previously reconstructed HR image in the overlapped region. Finally, the target HR image patch \mathbf{x}_i^p is obtained by multiplying the sparse coefficients $\boldsymbol{\alpha}_i^p$ with the HR dictionary \mathbf{D}_{h_patch} as follows:

$$\mathbf{x}_i^p = \mathbf{D}_{h_patch} \boldsymbol{\alpha}_i^p, \quad (4.7)$$

Next, by tiling all the reconstructed HR image patches, a patch-based HR image is first approximated and subsequently refined to obtain \mathbf{X}_{HR}^p by using the global constraint-based image reconstruction model of Eq. 3.16 in Chapter 3.

Further, the keypoints are detected from the resized upscaled LR image \mathbf{Y}_{IH} and \mathbf{X}_{HR}^p obtained above using the SIFT, and their matching keypoints are identified. Next, patches encircling the matched keypoints \mathbf{y}_i^k of \mathbf{Y}_{IH} are extracted in order to calculate the sparse coding $\boldsymbol{\alpha}_i^k$ using $\mathbf{D}_{\ell_keypoint}$ and reconstruct the HR patch \mathbf{x}_i^k using $\mathbf{x}_i^k = \mathbf{D}_{h_keypoint} \boldsymbol{\alpha}_i^k$. The original patches of \mathbf{X}_{HR}^p are then replaced by the keypoint driven patches in their respective co-ordinates in order to obtain the HR image \mathbf{X}_{HR}^k . Following the above, a keypoint-driven patch sparsity-based regularization problem may be defined as follows:

$$R_{KPSR}(\mathbf{X}_{HR}^k) = \min_{\boldsymbol{\alpha}_i^k} \|\mathbf{y}_i^k - \mathbf{D}_{\ell_keypoint} \boldsymbol{\alpha}_i^k\|_2^2 + \lambda \|\boldsymbol{\alpha}_i^k\|_1, \quad (4.8)$$

where $\mathbf{y}_i^k = \mathcal{K}_i(SH\mathbf{X}_{HR}^k)$ is a key-driven patch extracted using the operator \mathcal{K} from the i^{th} position. \mathcal{K} operator is the SIFT detector to identify matching keypoints from \mathbf{Y}_{IH} . It is efficiently solved using the feature-sign method [118]. Eqs. 4.6

and 4.8 use the same λ value as Eqs. 4.4 and 4.5. Since MS images contain many smooth regions and edges besides rich textural regions, the result obtained above is unable to restore both the sharp edges and smooth structures equally well as may be also observed in Fig. 4.4. In order to preserve both the textural and structural information of the reconstructed image simultaneously, a joint sparse reconstruction model involving two regularization priors, namely, the keypoints-driven patch-based sparse representation (KPSR) and the NLTV is proposed for the reconstruction phase. The second regularization prior term using the NLTV prior on \mathbf{X}_{HR}^k is defined as:

$$R_{NLTV}(\mathbf{X}_{HR}^k) = \|\mathbf{X}_{HR}^k - W\mathbf{X}_{HR}^k\|_F^2 = \sum_{i \in \mathbf{x}_{HR_i}^k} \|\mathbf{x}_{HR_i}^k - \mathbf{w}_i^T \mathbf{s}_i\|_2^2, \quad (4.9)$$

where for a particular pixel $\mathbf{x}_{HR_i}^k$ in \mathbf{X}_{HR}^k , a weighted average of neighboring pixels within a search window is obtained using \mathbf{s}_i and \mathbf{w}_i , where \mathbf{s}_i is the column vector that contains all of the central pixels in the search window around $\mathbf{x}_{HR_i}^k$ and \mathbf{w}_i is the column vector that contains the corresponding weights co-efficients w_{ij} . The calculation of w_{ij} is described in [126], and it is determined by the similarity between the pixel within the search window using the ℓ_2 -distance. An augmented Lagrangian method [126] is finally used to solve the above regularization subproblem efficiently. Now, by combining the two regularization priors, i.e. Eqs. 4.8 and 4.9, the proposed edge-preserving joint sparse model is formulated, which is defined as follows:

$$\begin{aligned} \hat{\mathbf{X}}_{HR}^k = \arg \min_{\mathbf{X}_{HR}^k} & \|SH\mathbf{X}_{HR}^k - \mathbf{Y}\|_F^2 + \mu_1 R_{KPSR}(\mathbf{X}_{HR}^k) \\ & + \mu_2 R_{NLTV}(\mathbf{X}_{HR}^k), \end{aligned} \quad (4.10)$$

where μ_1 and μ_2 are positive regularization parameters. Regularization subproblems corresponding to the second and third terms in the above equation are solved independently within the ADMM framework and then substitute their results into

Eq. 4.10 to obtain the final sub-problem, defined as follows:

$$\begin{aligned} \hat{\mathbf{X}}_{HR}^k = \arg \min_{\mathbf{X}_{HR}^k} & \frac{1}{2} \|\mathbf{SHX}_{HR}^k - \mathbf{Y}\|_F^2 + \\ & \frac{\mu_1}{2} \sum_i \left(\|\mathcal{K}_i(\mathbf{SHX}_{HR}^k) - \mathbf{D}_{\ell_keypoint} \boldsymbol{\alpha}_i^k\|_2^2 \right) + \frac{\mu_2}{2} \sum_{i \in \mathbf{x}_{HR_i}^k} \|\mathbf{x}_{HR_i}^k - \mathbf{w}_i^T \mathbf{s}_i\|_2^2. \end{aligned} \quad (4.11)$$

Algorithm 3: The proposed edge-preserving reconstruction algorithm

Input: \mathbf{Y} , \mathbf{D}_{ℓ_patch} , \mathbf{D}_{h_patch} , $\mathbf{D}_{\ell_keypoint}$, $\mathbf{D}_{h_keypoint}$

- 1 Initialization: $k \leftarrow 0$, $\delta \leftarrow 10^{-4}$, μ_1 , μ_2 ;
- 2 Upsample \mathbf{Y} to \mathbf{Y}_{IH} using Lanczos interpolation.
- 3 Extract four high-frequency feature maps from \mathbf{Y}_{IH} : Sobel in x-, and y-directions, DoG, Butterworth.
- 4 **for** each LR patch $\mathbf{y}_{\ell_i}^p$ of \mathbf{Y}_{IH} **do**
- 5 Calculate sparse coefficients vector $\boldsymbol{\alpha}_i^p$ by:

$$\boldsymbol{\alpha}_i^p = \min_{\boldsymbol{\alpha}_i^p} \|\tilde{\mathbf{D}}_{c_patch} \boldsymbol{\alpha}_i^p - \tilde{\mathbf{y}}_i^p\|_2^2 + \lambda \|\boldsymbol{\alpha}_i^p\|_1$$
- 6 Generate HR image patch $\mathbf{x}_i^p = \mathbf{D}_{h_patch} \boldsymbol{\alpha}_i^p$
- 7 **end**
- 8 Tile reconstructed HR patches to approximate \mathbf{X}_{HR}^p . Refine \mathbf{X}_{HR}^p by applying global constraint-based reconstruction model
- 9 Find the matching keypoints between \mathbf{Y}_{IH} and \mathbf{X}_{HR}^p
- 10 **for** each patch encircling the matched keypoints \mathbf{y}_i^k of \mathbf{Y}_{IH} **do**
- 11 Calculate Sparse coding $\boldsymbol{\alpha}_i^k$ using $\mathbf{D}_{\ell_keypoint}$ and \mathbf{y}_i^k
- 12 Generate HR keypoint driven patch $\mathbf{x}_i^k = \mathbf{D}_{h_keypoint} \boldsymbol{\alpha}_i^k$
- 13 **end**
- 14 Replace patches of \mathbf{X}_{HR}^p with keypoint-driven patches to obtain \mathbf{X}_{HR}^k
- 15 **while** not converge **do**
- 16 $k \leftarrow k+1$
- 17 Define keypoint-driven patch sparsity-based regularization problem:

$$R_{KPSR}(\mathbf{X}_{HR}^k) = \min_{\boldsymbol{\alpha}_i^k} \|\mathbf{y}_i^k - \mathbf{D}_{\ell_keypoint} \boldsymbol{\alpha}_i^k\|_2^2 + \lambda \|\boldsymbol{\alpha}_i^k\|_1$$
- 18 Define NLTV prior-based regularization problem:

$$R_{NLTV}(\mathbf{X}_{HR}^k) = \|\mathbf{X}_{HR}^k - \mathbf{W} \mathbf{X}_{HR}^k\|_F^2 = \sum_{i \in \mathbf{x}_{HR_i}^k} \|\mathbf{x}_{HR_i}^k - \mathbf{w}_i^T \mathbf{s}_i\|_2^2$$
- 19 Using ADMM to solve the joint sparse model:

$$\hat{\mathbf{X}}_{HR}^k = \arg \min_{\mathbf{X}_{HR}^k} \frac{1}{2} \|\mathbf{SHX}_{HR}^k - \mathbf{Y}\|_F^2 + \frac{\mu_1}{2} \sum_i \left(\|\mathcal{K}_i(\mathbf{SHX}_{HR}^k) - \mathbf{D}_{\ell_keypoint} \boldsymbol{\alpha}_i^k\|_2^2 \right) + \frac{\mu_2}{2} \sum_{i \in \mathbf{x}_{HR_i}^k} \|\mathbf{x}_{HR_i}^k - \mathbf{w}_i^T \mathbf{s}_i\|_2^2$$
- 20 Check convergence: $\left\| \hat{\mathbf{X}}_{HR}^k - \hat{\mathbf{X}}_{HR}^{k-1} \right\| / \left\| \hat{\mathbf{X}}_{HR}^k \right\| \leq \delta$
- 21 **end**

Output: $\hat{\mathbf{X}}_{HR}$

It is worth mentioning that the performance of the above joint sparse model depends significantly on the values of μ_1 and μ_2 . In Section 4.4.6, an approach for selection of optimal values of μ_1 and μ_2 are demonstrated through an experiment. The above problem is a simple least-square convex minimization problem, which can be solved in a close form by differentiating it w.r.t. \mathbf{X}_{HR}^k . Conjugate gradient least square solver (CGLS) is used to solve the above subproblem as detailed in [13, 21]. The proposed edge-preserving reconstruction algorithm is presented in Algorithm 3.

4.3.4 Implementation using CUDA-GPU

In this section, implementation of the proposed method on a CUDA-enabled GPU platform using the dedicated cuBLAS and cuSPARSE libraries and user defined kernels exploiting thread-level parallelism is detailed. It gives a fair idea on the overall GPU-based acceleration that could be achieved on the proposed scheme.

Algorithm 4: CUDA-GPU based BOMP.

Data:
D: Coupled dictionary
Y: The $n \times m$ patch matrix. Each column is a separate patch vector we want to sparse-encode
D_t: precomputed $D^\top D$
D_y: precomputed $D^\top Y$
K: target sparsity

Procedure:

- 1 Define kernel function by using `_global_` specifier to sparse-encode the patch vectors separately.
`int j = blockIdx.x * blockDim.x + threadIdx.x`
- 2 **if** $j \geq m - 1$ **then**
- 3 | return;
- 4 **end**
- 5 **for** $i \leftarrow 1$ **to** m **do**
- 6 | $\alpha[j + i * m] = \text{BOMP}(\mathbf{D}_t, \mathbf{D}_y[:, i], K)$;
- 7 **end**

Result: α : Sparse representation

4.3.4.1 CUDA GPU-based multiple coupled dictionary learning

Both patch-based- and keypoints-based dictionaries employ the KSVD algorithm for dictionary learning. However, it is worth noting that K-SVD dictionary learning is a highly iterative and time consuming process. In each iteration, the atoms of the dictionary is updated one-by-one, taking K-singular value decompositions (SVD) for

the entire dictionary. Another time-expensive iterative process is the sparse coding using the OMP. To mitigate this challenge, the KSVD process is accelerated by utilizing GPU-CUDA, harnessing the immense computational power of GPUs for faster execution.

In the proposed algorithm, we use the batch OMP (BOMP) for sparse coding as it speeds up the convergence considerably [2]. Further, we carry out a parallel implementation of the BOMP on CUDA platform by using interleaved arrays. It is implemented as $\alpha[j + i * m]$, where m is the number of active threads and $j = blockIdx.x * blockDim.x + threadIdx.x$ is the index of the current CUDA thread. Each thread solves a separate BOMP problem. A pseudo-code representation of the CUDA GPU-based BOMP implementation is shown in Algorithm 4. In particular, “cublasIdamax” and “cublasSgemm” functions from cuBLAS API are used for finding the index with the largest projection and updating the provisional solution, respectively.

Besides, KSVD consists of many other linear algebraic operations, particularly matrix operations; several on-device routines are borrowed from the standard cuBLAS library to speed up the execution of these operations. GPU kernel implementations based on thread-level parallelism are also done in cases where mathematical operations are to be performed massively on a row-by-row or column-by-column basis in parallel. For example, a kernel is specified for parallelly collecting non-zero elements from j number of rows of a sparse matrix into a dense array. The schematic diagram of CUDA-GPU KSVD is shown in Fig. 4.2a.

4.3.4.2 CUDA-based SR reconstruction on GPU

In the reconstruction, resized input LR image is divided into multiple overlapping feature patches, which are then processed independently. The processing time increases as the image size increases. In this scenario, we deploy the GPU environment using the CUDA programming model for accelerating the reconstruction time. The proposed scheme is graphically shown in Fig. 4.2b. First, LR feature patches, pre-

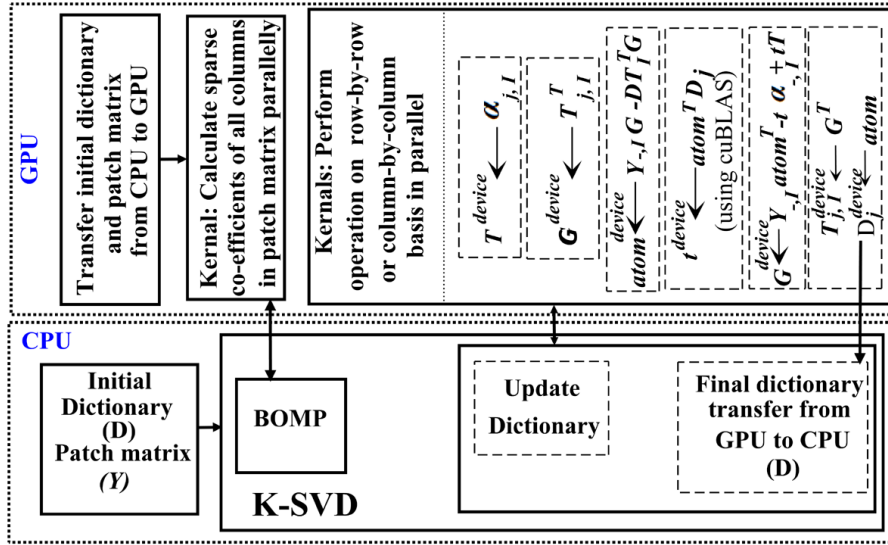
trained patch-based coupled dictionaries, pre-trained keypoint driven LR patches, and pretrained keypoint driven coupled dictionaries are transferred from the CPU to the GPU. As shown in stage I, a CUDA kernel function is defined, where the feature image patches are to be processed by different GPU blocks and then patches are distributed across the threads within each GPU block. Another kernel is specified for estimating the sparse co-efficients for each LR patch. Since it also involves a number of linear algebra operations, such as matrix-matrix multiplication, matrix-vector multiplication, and matrix-inverse, they are implemented using the cuBLAS functions to speed up the algorithms. Subsequently, HR patches are obtained parallelly by using kernel that multiply the \mathbf{D}_{h_patch} and $\boldsymbol{\alpha}^p$ corresponding to each patch by using the “cublasSgemv” function. Further, a kernel is specified for accumulating all the HR patches from all the active threads to reconstruct \mathbf{X}_{HR}^p . In stage-II, a CUDA kernel is used to assign all keypoint-driven LR patches into individual threads. Next, steps for the calculation of $\boldsymbol{\alpha}^k$ for keypoint-driven LR patch and reconstruction of the corresponding HR are implemented using similar CUDA kernels, as mentioned in stage-I. HR patches of \mathbf{X}_{HR}^p are then replaced by keypoint-driven patches parallelly using CUDA kernel. Finally, transfer of the keypoint-driven HR image \mathbf{X}_{HR}^k is done from GPU to CPU .

In ADMM framework, in order to speed-up the CGLS algorithm, “cusparseDcsrmmv” function from CUSPARSE library is used to implement sparse matrix multiplications, while a CUDA kernel is used to implement diagonal addition of matrices. Some cuBLAS functions are also utilised for acceleration in the CGLS method.

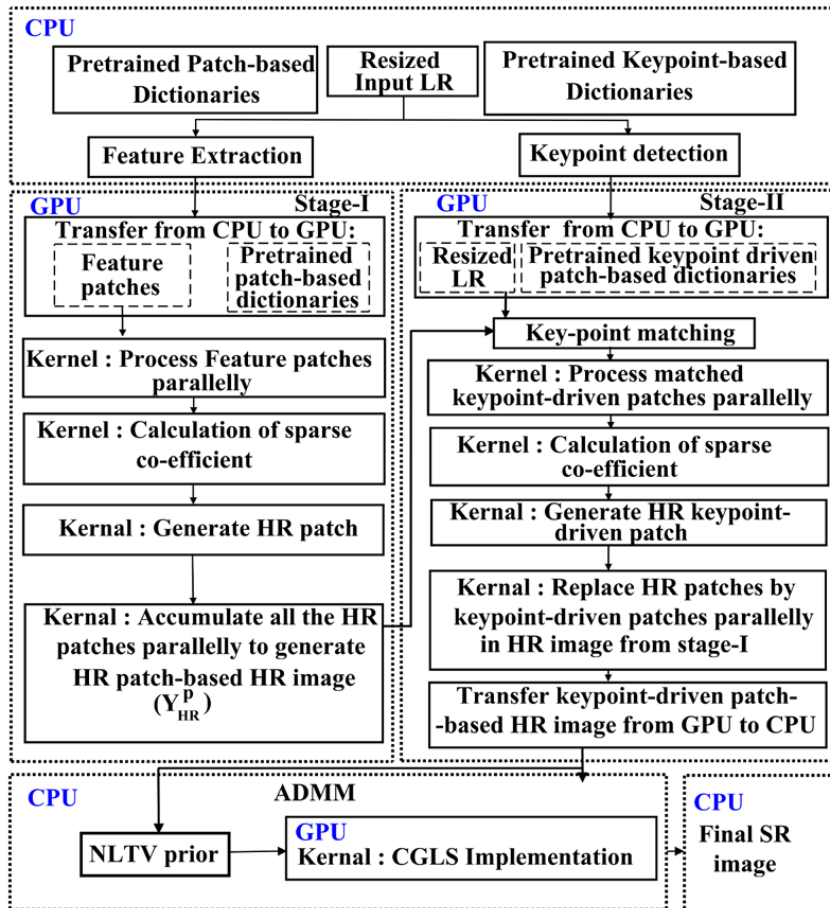
4.4 Results and Discussion

4.4.1 Computing environment

Simulations are carried out on a Ubuntu server running 16.04 LTS and equipped with Intel[®] Xeon[®] processor and having 128 GB RAM. C++ programming with



(a) CUDA GPU-accelerated KSVD algorithm used in patch- and keypoints-based multiple dictionaries



(b) CUDA GPU-based SR Reconstruction

Figure 4.2: CUDA-accelerated feature-enhanced KSVD algorithm and SR reconstruction.

OpenCV libraries (version 3.3.1) are used for CPU-based implementations. While GPU implementations are done on the NVIDIA Tesla P100 GP-GPU hardware platform with CUDA toolkit (version 9.0). DL-based SR methods used for comparisons are implemented using Python 3.6.4 and CUDA toolkit version 11.2.

4.4.1.1 Datasets preparation and parameter selection

RS images from two publicly available datasets (aerial image dataset (AID) [117] and PatternNet [138]) and two self-procured MS remote sensing datasets (LISS-IV¹ and LISS-III¹) are used for the experiments. AID comprises of 30 distinct classes with a total of 10,000 images, each of size 600×600 with a sensor spatial resolution ranging from 0.5 m to 0.8 m. Similarly, PatternNet has 38 classes; each class having 800 images at a spatial resolution of 0.062 m to 4.693 m and each of size 256×256 . For training the sparse representations-based SR methods, a total of 300 and 380 images are randomly chosen from the AID and the PatternNet, respectively. Since each class (both in AID and PatternNet) belongs to images of many similar patches, we chose only 5 to 10 images for each class of AID and PatternNet datasets. On the other hand, for the DL-based SR networks training, we randomly select 70-80 % of the total data from both the datasets in such a way that the selected images contain both rich textural and structural information, each having a size of 256×256 .

Table 4.2: Specifications of LISS-IV and LISS-III satellite sensors.

Satellite Sensor	LISS-IV	LISS-III
Spatial resolution	5.8 m	23.5 m
Spectral bands	Green (Band 2: 0.52-0.59 μm) Red (Band 3: 0.62-0.68 μm) Near infrared (Band 4: 0.77-0.86 μm)	Green (Band 2: 0.52-0.59 μm) Red (Band 3: 0.62-0.68 μm) Near infrared (Band 4: 0.77-0.86 μm) Mid infrared (Band 5: 1.55-1.70 μm)
Swath	23.9 km	141 km
Image size (in pixels)	18000×16000	7700×7000

Furthermore, the specifications of LISS-III and LISS-IV sensors are given in Table 4.2. To select test images from these datasets for dictionary training, 200 regions of interests (RoIs) ranging from 256×256 to 512×512 are cropped. In a similar way, in the DL-based training, an equal number of RoIs of the size 256×256

¹NRSC Data Center: <https://www.nrsc.gov.in/eos> dissemination

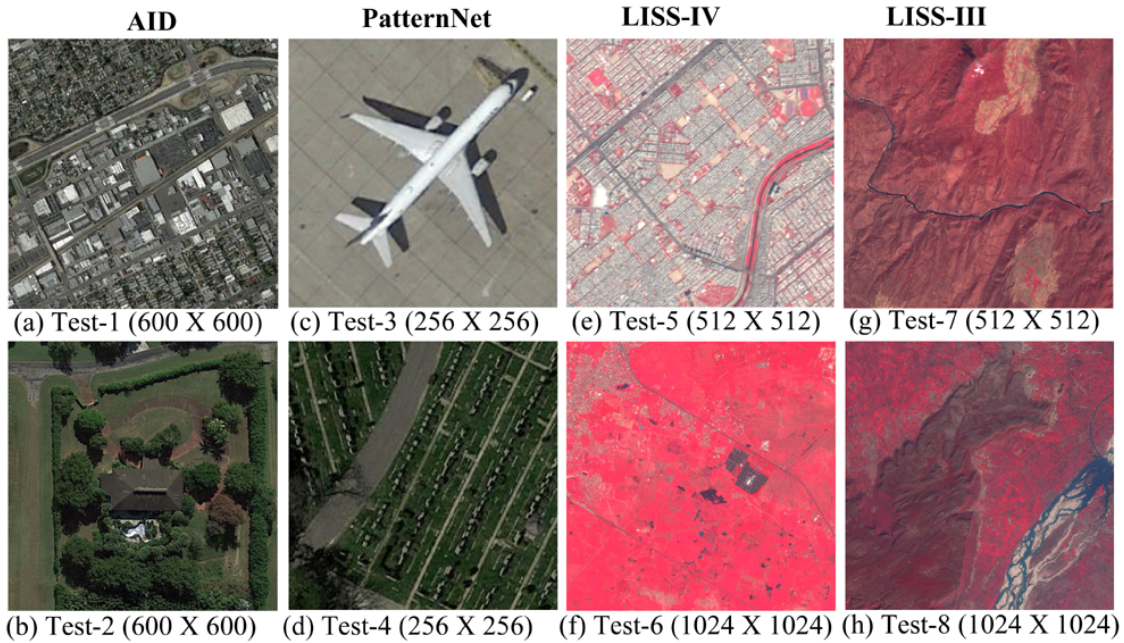


Figure 4.3: Representative HR images of different datasets considered for testing the proposed method and other state-of-the-art SR methods. Column wise: Test 1 and Test 2 from AID; Test 3 and Test 4 from PatternNet; Test 5 and Test 6 from LISS-IV; Test 7 and Test 8 from LISS-III.

are to be cropped from the original images of LISS-III and LISS-IV sensors to bring uniformity in terms of number and size of images with respect to the two benchmark DL datasets (AID, and PatternNet). Two representative test images from each dataset are used to check the effectiveness of the proposed method over all the compared SR methods as shown in Fig. 4.3. For testing, original LR images are blurred by using 5×5 ($\sigma = 1.2$), 7×7 ($\sigma = 1.4$), and 9×9 ($\sigma = 1.6$) Gaussian filters followed by downsampling by 2, 3 and 4, respectively. All the simulations are performed with the following parameter settings: dictionary size: 512, patch size: 7×7 , no. of pixels in overlap for patch extraction: 6, regularization parameters: $\lambda = 0.15$, constants: $\mu_1 = 0.00015$, $\mu_2 = 0.005$ and $\delta = 10^{-4}$.

4.4.2 Ablation Study

An ablation study is also conducted to validate the effectiveness of each component of the proposed method on AID test dataset for $2\times$ factor as shown in Fig.4.4. Specifically, the study evaluates the impact of different components, showing that

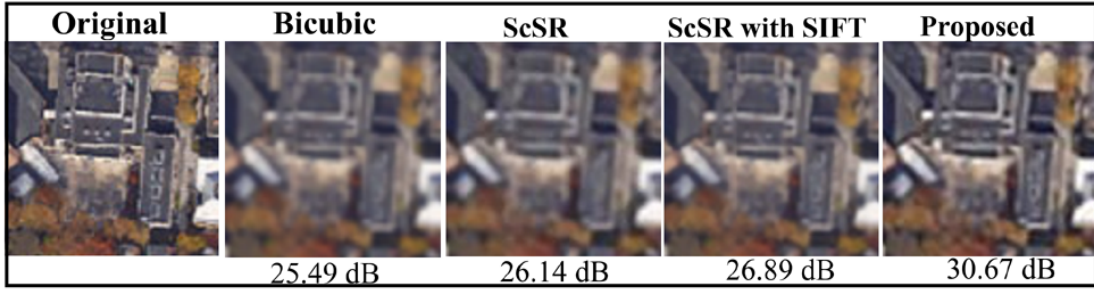


Figure 4.4: Evaluation of the impact of each component of the proposed method on the SR performance.

using only patch-based reconstruction (ScSR) i.e. $\arg \min_{\mathbf{X}_{HR}^k} \frac{1}{2} \|SH\mathbf{X}_{HR}^k - \mathbf{Y}\|_F^2$ yields a PSNR of 26.14 dB. However, incorporating SIFT keypoints-driven reconstruction alongside patch-based methods (ScSR with SIFT) i.e. $\arg \min_{\mathbf{X}_{HR}^k} \|SH\mathbf{X}_{HR}^k - \mathbf{Y}\|_F^2 + \|\mathcal{K}_i(SH\mathbf{X}_{HR}^k) - \mathbf{D}_{\ell_keypoint}\boldsymbol{\alpha}_i^k\|_2^2$ enhances PSNR by an additional 0.75 dB. Moreover, the proposed joint reconstruction model results in PSNR improvements of 4.53 dB over patch-based methods.

4.4.3 Performance evaluation

We have compared the proposed method both visually and quantitatively with state-of-the-art methods: sparse-representations-based SR methods, like ScSR [118], A+ [99], CCR [131], JRSR [14], CDLSR [90], and DL-based SR methods, like SRCNN [22], VDSR [45], MHAN [125], SAN [18] and CFSRCNN [97]. Visual results are analyzed to assess the perceptual quality of the reconstructed images. Furthermore, quantitative evaluations are carried out in terms of PSNR, SSIM, ERGAS [105], SAM [121], Q-index [113] and sCC [137], which are commonly used for the evaluation of the quality of RS image reconstruction. Besides, EPI [84] and PSI [30] metrics are used to measure the quantity of edges preservation, and sharpness in the reconstructed image. Higher PSNR, SSIM, UIQI, sCC, EPI and PSI and lower ERGAS and SAM indicate that the reconstructed image quality is better.

4.4.3.1 Visual results

A visual comparison of various SR methods is performed on ‘Test-2’ and ‘Test-5’ images for different zooming factors. The reconstructed images for different scaling factors are shown in Fig. 4.5 and 4.6 for ‘Test-2’ and ‘Test-5’, respectively. All the results are shown in false RGB format as the test images are in false color composition only. In the figure, ROIs are highlighted and presented in their scaled



Figure 4.5: Visual results of different methods on Test-2 of AID for different zooming factors.

versions for better interpretation of the visual results. It is clear from the results that the perceptual quality of the proposed method is better than those of other methods. It is capable of retaining textural as well as structural characteristics

with sharp edges. Other SR approaches, such as ScSR, A+, CCR, CDLSR, and DL-methods do not consider blurring in the degradation model to generate the LR images in their works; they only use downsampling operation. It is clearly visible

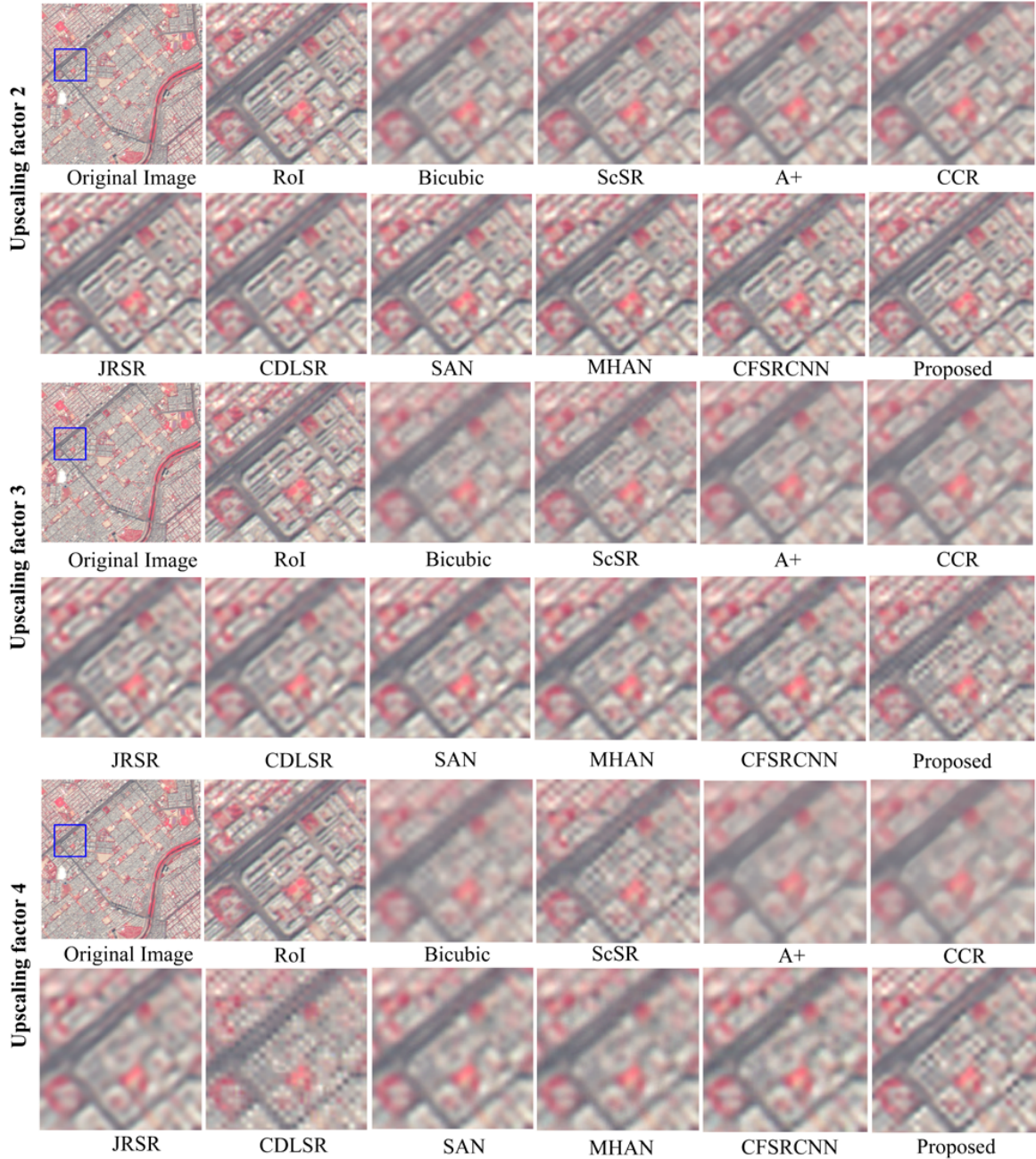


Figure 4.6: Visual results of different methods on Test-5 of LISS-IV for different zooming factors.

that these methods are unable to overcome the effects of blurring and downsampling, simultaneously. Although visual outputs of these methods for $2\times$ show satisfactory appearance to some extent, their qualities suffer greatly for $3\times$ and $4\times$ upscaling.

Furthermore, the visual output of JRSR is slightly closer to that of the proposed method for $2\times$ zooming factor, but it produces smoother results for higher zooming factors, such as $3\times$ and $4\times$. While, CDLSR produces block artifacts around the edges for $4\times$ zooming factor. However, the proposed method provides better visual output in terms of edges and other structures for different zooming factors ($2\times$, $3\times$ and $4\times$).

4.4.3.2 Quantitative analysis

Quantitative analysis of several state-of-the-art SR methods in terms of different objective evaluation metrics are shown in Tables 4.4–4.6 using different datasets for different zooming factors. In case of AID, the proposed method has the highest average PSNR, which results in an improvement of 1.2–3.5 dB, 1.7–4.1 dB, and 2.1–4 dB over other methods for $\times 2$, $\times 3$, and $\times 4$ zooming, respectively, as shown in Table 4.3. From Table 4.4, it can be observed that the proposed method achieves the maximum PSNR for PatternNet, with improvements of 2.1–5.8 dB, 1.4–6.4 dB, and 0.66–5.3 dB for $\times 2$, $\times 3$, and $\times 4$ zooming, respectively. Although the proposed method performs well on PatternNet, its improvement is lower than that of the AID test data for $\times 4$ zooming. Therefore, it is observed that the proposed method underperforms on images with less edges and structures. Similarly, SSIM, ERGAS, SAM, Q-Index, sCC values of the proposed method outperform others for $\times 2$, $\times 3$, and $\times 4$ zooming on both the datasets. Tables 4.4 and 4.3 clearly show that EPI and PSI scores of the proposed method are greater than the other SR methods for all the test data of Patternnet and AID. Therefore, it is very much evident that the efficiency of proposed method to restore the edges and sharp structure is highly promising. Although JRSR provides comparable PSNR results, it performs poorly in terms of EPI and PSI compared to the proposed method. Further, the proposed method is also compared with some of the state-of-the-art DL-based SR methods, i.e. SRCNN, VDSR, SAN, CFSRCNN and MHAN. It is found that the proposed method performs better than the DL-based SR methods. While, this observation is rather surprising for the author as such, the reason may be probably due to the blurring

Chapter 4. Development of Edge Preserving Remote Sensing Single Image Super-resolution based on Global Dictionary Learning and Sparse Representations

Table 4.4: Quantitative analysis of different methods on the test images of PatternNet for 2x, 3x and 4x zooming.

Zooming Factor	Test Image	Metric	PatternNet															
			Bicubic	sCSR	A+	CCR	JRSR	CDLSR	SRcNN	VDSR	SAN	MHAN	CSRCNN	Proposed				
x2	Test 3	PSNR (dB) ↑	30.94	31.76	33.86	33.83	35.58	34.02	31.84	32.23	35.16	35.21	35.18	37.29				
		SSIM ↑	0.893	0.921	0.923	0.883	0.937	0.883	0.911	0.912	0.913	0.912	0.913	0.948				
		ERGAS ↓ [105]	2.614	2.377	1.869	1.875	1.536	1.835	2.362	2.256	1.612	1.603	1.609	1.263				
		SAM ↓ [121]	0.0508	0.0462	0.0362	0.0364	0.0297	0.0356	0.0458	0.0437	0.0312	0.0310	0.0312	0.0244				
		UIQI ↑ [113]	0.6717	0.7790	0.7596	0.7598	0.7985	0.5831	0.5812	0.7255	0.6496	0.6566	0.6566	0.8231				
		sCC ↑ [137]	0.9746	0.9791	0.9881	0.9881	0.9912	0.9878	0.9795	0.9812	0.9903	0.9905	0.9904	0.9941				
		EPI ↑ [84]	0.6924	0.7405	0.7501	0.7505	0.7202	0.7680	0.7223	0.7235	0.7063	0.7569	0.7566	0.7908				
		PSI* ↑ [84]	0.1726	0.1883	0.1892	0.1833	0.2733	0.2921	0.2699	0.2556	0.2536	0.2514	0.2630	0.2563				
x3	Test 3	PSNR (dB) ↑	27.65	27.47	32.08	32.10	34.85	34.57	29.35	30.02	32.30	32.49	32.38	36.20				
		SSIM ↑	0.846	0.850	0.899	0.898	0.918	0.916	0.836	0.858	0.869	0.873	0.871	0.945				
		ERGAS ↓ [105]	3.820	3.899	2.294	2.288	1.669	1.724	3.139	2.908	2.239	2.191	2.218	1.429				
		SAM ↓ [121]	0.0743	0.0759	0.0445	0.0444	0.0324	0.0334	0.0609	0.0565	0.0432	0.0425	0.0431	0.0278				
		UIQI ↑ [113]	0.5735	0.5960	0.6941	0.6898	0.7531	0.7445	0.4252	0.4782	0.4780	0.4873	0.4886	0.8361				
		sCC ↑ [137]	0.9448	0.9422	0.9826	0.9826	0.9897	0.9890	0.9636	0.9730	0.9822	0.9816	0.9817	0.9924				
		EPI ↑ [84]	0.5543	0.5881	0.7050	0.7020	0.7289	0.7289	0.6285	0.6815	0.6826	0.6894	0.6917	0.7501				
		PSI ↑ [30]	0.1544	0.1629	0.1672	0.1773	0.2349	0.2661	0.2020	0.2585	0.2505	0.2514	0.2555	0.2784				
x4	Test 3	PSNR (dB) ↑	25.49	25.22	30.33	30.32	31.97	31.81	28.41	29.04	29.55	30.31	30.22	32.22				
		SSIM ↑	0.7831	0.7888	0.8598	0.8588	0.8634	0.8634	0.8022	0.8170	0.8191	0.8336	0.8321	0.8913				
		ERGAS ↓ [105]	4.895	5.048	2.804	2.808	2.326	2.370	3.500	3.260	2.814	3.070	2.844	2.260				
		SAM ↓ [121]	0.0954	0.0984	0.0544	0.0545	0.0451	0.0460	0.0680	0.0631	0.0596	0.0547	0.0552	0.0439				
		UIQI ↑ [113]	0.3878	0.4431	0.5735	0.5717	0.6150	0.6050	0.3424	0.3371	0.3624	0.3624	0.3644	0.6924				
		sCC ↑ [137]	0.9065	0.9008	0.9738	0.9736	0.9799	0.9790	0.9541	0.9604	0.9645	0.9704	0.9697	0.9809				
		EPI ↑ [84]	0.4486	0.4522	0.6021	0.6023	0.6336	0.6278	0.6101	0.6295	0.6376	0.6343	0.6343	0.7057				
		PSI ↑ [30]	0.1233	0.1431	0.1671	0.1567	0.1712	0.1815	0.1700	0.1673	0.1760	0.2232	0.1902	0.2211				
x2	Test 4	PSNR (dB) ↑	20.03	20.65	22.03	22.04	22.73	23.98	21.99	22.33	22.47	23.74	23.67	25.26				
		SSIM ↑	0.621	0.684	0.694	0.694	0.731	0.822	0.669	0.665	0.737	0.757	0.754	0.875				
		ERGAS ↓ [105]	16.960	15.805	14.666	14.644	13.532	11.192	15.232	15.300	13.311	12.672	12.771	9.320				
		SAM ↓ [121]	0.2860	0.2658	0.2457	0.2454	0.2271	0.1871	0.2562	0.2574	0.2123	0.2233	0.2140	0.1543				
		UIQI ↑ [113]	0.5744	0.6495	0.6649	0.6659	0.7142	0.8086	0.6339	0.6284	0.7160	0.7247	0.7233	0.8542				
		sCC ↑ [137]	0.8404	0.8634	0.8866	0.8868	0.9000	0.9330	0.8712	0.8699	0.9036	0.9130	0.9114	0.9572				
		EPI ↑ [84]	0.5093	0.5299	0.6022	0.6002	0.5522	0.6262	0.5249	0.5271	0.5891	0.6163	0.6160	0.7698				
		PSI ↑ [30]	0.2470	0.2510	0.2714	0.2538	0.3118	0.3542	0.3083	0.3083	0.3330	0.3366	0.3423	0.4198				
x3	Test 4	PSNR ↑	18.92	18.66	20.87	20.87	21.46	21.31	19.47	21.95	20.61	20.61	20.56	22.85				
		SSIM ↑	0.468	0.472	0.578	0.578	0.662	0.652	0.487	0.593	0.584	0.587	0.583	0.715				
		ERGAS ↓ [105]	20.731	21.355	17.200	17.196	15.459	15.727	19.461	18.813	17.180	17.076	17.168	14.213				
		SAM ↓ [121]	0.3522	0.3633	0.2906	0.2906	0.2602	0.2648	0.3294	0.3148	0.2878	0.2897	0.2895	0.2287				
		UIQI ↑ [113]	0.3907	0.4061	0.5352	0.5356	0.6447	0.6305	0.4102	0.5502	0.5291	0.5328	0.5235	0.7032				
		sCC ↑ [137]	0.7428	0.7248	0.8336	0.8336	0.8664	0.8614	0.7783	0.8184	0.8321	0.8343	0.8323	0.8983				
		EPI ↑ [84]	0.4052	0.4184	0.4921	0.4952	0.5027	0.5032	0.4313	0.4886	0.4942	0.4975	0.5041	0.5366				
		PSI ↑ [30]	0.1923	0.2060	0.2161	0.2422	0.2579	0.2621	0.2370	0.2566	0.2553	0.3366	0.2725	0.3354				
x4	Test 4	PSNR ↑	17.334	17.07	19.70	19.72	19.25	19.18	18.07	18.12	18.442	18.58	18.54	20.33				
		SSIM ↑	0.368	0.378	0.486	0.488	0.524	0.516	0.407	0.410	0.443	0.456	0.451	0.611				
		ERGAS ↓ [105]	23.170	23.889	19.234	19.173	18.582	18.724	21.278	21.161	20.405	20.073	20.171	18.017				
		SAM ↓ [121]	0.3962	0.4086	0.3266	0.3255	0.3145	0.3170	0.3622	0.3600	0.3406	0.3465	0.3423	0.2996				
		UIQI ↑ [113]	0.2451	0.2725	0.4149	0.4144	0.4776	0.4671	0.2948	0.2953	0.3461	0.3605	0.3522	0.5346				
		sCC ↑ [137]	0.6645	0.6451	0.7822	0.7839	0.8007	0.7972	0.7273	0.7310	0.7533	0.7625	0.7599	0.8243				
		EPI ↑ [84]	0.3905	0.4088	0.4035	0.4164	0.3963	0.3976	0.4518	0.4222	0.4298	0.4221	0.4154	0.4754				
		PSI ↑ [30]	0.1945	0.1987	0.2090	0.2237	0.2086	0.2022	0.1833	0.1837	0.1885	0.2146	0.2228	0.2852				

'↑' means the higher the value, the better and '↓' denotes that the smaller the value is, the better. Bold: indicates the best results.

Table 4.5: Performance evaluation of different methods for test images of LISS-IV using $\times 2$, $\times 3$ and $\times 4$ zooming.

Zooming Factor	TestImage	Metric	LISS-IV										Proposed
			Bicubic	ScSR	A+	CCR	JRSR	CDLSSR	SAN	MHAN	CSRCNN	CSRCNN	
$\times 2$	Test 5	PSNR (dB) ↑	25.36	26.01	27.34	27.34	30.30	28.61	29.08	29.62	28.98	31.46	
		SSIM ↑	0.808	0.840	0.882	0.882	0.958	0.922	0.948	0.953	0.946	0.973	
		ERGAS ↓ [105]	3.794	3.604	3.088	3.091	2.192	2.666	1.919	2.373	2.533	1.756	
		SAM ↓ [121]	0.0749	0.0712	0.0609	0.0610	0.0433	0.0526	0.0498	0.0468	0.0501	0.0378	
		UIQI ↑ [113]	0.4370	0.4556	0.6037	0.6032	0.6636	0.6412	0.6633	0.6599	0.6694	0.7086	
		sCC ↑ [137]	0.8803	0.8921	0.9272	0.9270	0.9604	0.9410	0.9472	0.9536	0.9465	0.9708	
		EPI ↑ [84]	0.6780	0.6813	0.8705	0.8580	0.7414	0.6634	0.7374	0.7703	0.7831	0.8067	
		PSI ↑ [30]	0.2364	0.2413	0.2497	0.2589	0.3410	0.3153	0.3048	0.3027	0.3129	0.3715	
		PSNR (dB) ↑	23.39	23.19	25.41	25.40	25.88	25.61	25.85	25.90	25.87	27.20	
		SSIM ↑	0.631	0.641	0.775	0.776	0.820	0.804	0.822	0.819	0.827	0.868	
		ERGAS ↓ [105]	4.874	4.988	3.862	3.867	3.651	3.768	3.671	3.647	3.659	3.241	
		SAM ↓ [121]	0.0964	0.0986	0.0763	0.0764	0.0721	0.0744	0.0725	0.0720	0.0718	0.0679	
UIQI ↑ [113]	0.4370	0.4556	0.6037	0.6032	0.6636	0.6412	0.6633	0.6599	0.6694	0.7090			
sCC ↑ [137]	0.7874	0.7758	0.8782	0.8777	0.8865	0.8787	0.8851	0.8865	0.8861	0.8999			
EPI ↑ [84]	0.3055	0.4029	0.5318	0.5204	0.5507	0.5515	0.5719	0.5767	0.5172	0.5812			
PSI ↑ [30]	0.2041	0.2107	0.2193	0.2229	0.2353	0.2260	0.2249	0.2336	0.2558	0.3050			
$\times 3$	Test 5	PSNR (dB) ↑	22.30	21.83	23.89	23.87	24.52	24.31	24.10	24.29	24.33	25.94	
		SSIM ↑	0.504	0.499	0.642	0.641	0.726	0.717	0.698	0.706	0.724	0.790	
		ERGAS ↓ [105]	5.536	5.833	4.600	4.614	4.277	4.383	4.491	4.396	4.374	3.986	
		SAM ↓ [121]	0.1093	0.1154	0.0909	0.0912	0.0845	0.0866	0.0889	0.0875	0.0883	0.0801	
		UIQI ↑ [113]	0.2972	0.3014	0.4311	0.4290	0.5344	0.5323	0.4929	0.5038	0.5277	0.6013	
		sCC ↑ [137]	0.7151	0.6830	0.8197	0.8180	0.8409	0.8333	0.8221	0.8304	0.8328	0.8580	
		EPI ↑ [84]	0.303	0.3368	0.3639	0.3676	0.4884	0.3872	0.4174	0.4479	0.4449	0.4893	
		PSI ↑ [30]	0.1557	0.1675	0.1730	0.1789	0.1906	0.2602	0.2081	0.1803	0.1804	0.2619	
		PSNR (dB) ↑	32.60	31.61	34.05	34.01	37.08	36.53	36.11	36.86	36.20	39.68	
		SSIM ↑	0.976	0.966	0.989	0.988	0.997	0.996	0.994	0.994	0.995	0.999	
		ERGAS ↓ [105]	2.574	2.885	2.176	2.188	1.460	1.636	1.723	1.583	1.702	1.164	
		SAM ↓ [121]	0.0505	0.0565	0.0426	0.0428	0.0266	0.0320	0.0335	0.0310	0.0333	0.0237	
UIQI ↑ [113]	0.6886	0.6034	0.7439	0.7305	0.8369	0.7954	0.7857	0.7964	0.7944	0.8360			
sCC ↑ [137]	0.9493	0.9365	0.9652	0.9647	0.9857	0.9795	0.9773	0.9810	0.9777	0.9978			
EPI ↑ [84]	0.5443	0.5629	0.6320	0.6339	0.7328	0.6802	0.7005	0.7271	0.6348	0.7783			
PSI ↑ [30]	0.1974	0.2097	0.2351	0.2337	0.3128	0.2875	0.2846	0.2880	0.2519	0.3298			
$\times 2$	Test 6	PSNR (dB) ↑	30.08	30.14	32.45	32.44	32.80	32.70	32.49	32.79	32.76	34.33	
		SSIM ↑	0.939	0.943	0.982	0.981	0.990	0.988	0.982	0.982	0.984	0.994	
		ERGAS ↓ [105]	3.4401	3.4218	2.627	2.625	2.466	2.562	2.621	2.545	2.538	2.275	
		SAM ↓ [121]	0.0675	0.0671	0.0515	0.0514	0.0483	0.0502	0.0509	0.0499	0.0501	0.0455	
		UIQI ↑ [113]	0.5142	0.5266	0.6463	0.6384	0.6565	0.6328	0.6256	0.6290	0.6361	0.6729	
		sCC ↑ [137]	0.9059	0.9065	0.9478	0.9478	0.9516	0.9492	0.9474	0.9503	0.9500	0.9669	
		EPI ↑ [84]	0.4295	0.4333	0.4706	0.4725	0.5133	0.4855	0.5451	0.4886	0.4502	0.5502	
		PSI ↑ [30]	0.2066	0.2107	0.2126	0.2153	0.2244	0.2197	0.2262	0.2306	0.2384	0.2836	
		PSNR (dB) ↑	28.72	28.38	30.76	30.74	31.01	31.31	30.78	30.79	30.59	32.67	
		SSIM ↑	0.913	0.923	0.960	0.959	0.987	0.974	0.960	0.957	0.958	0.993	
		ERGAS ↓ [105]	4.019	4.185	3.190	3.193	3.094	3.005	3.205	3.269	3.269	2.787	
		SAM ↓ [121]	0.0789	0.0822	0.0625	0.0626	0.0606	0.0589	0.0626	0.0628	0.0640	0.0556	
UIQI ↑ [113]	0.3876	0.3817	0.5118	0.5036	0.5176	0.5424	0.5031	0.4874	0.4901	0.6169			
sCC ↑ [137]	0.8681	0.8570	0.9210	0.9207	0.9253	0.9292	0.9195	0.9198	0.9157	0.9450			
EPI ↑ [84]	0.3442	0.3690	0.3582	0.3567	0.4048	0.4253	0.3493	0.3844	0.3978	0.4750			
PSI ↑ [30]	0.1806	0.1842	0.1856	0.1854	0.2401	0.1827	0.1766	0.1798	0.1849	0.2353			
$\times 4$	Test 6	PSNR (dB) ↑	28.72	28.38	30.76	30.74	31.01	31.31	30.78	30.79	30.59	32.67	
		SSIM ↑	0.913	0.923	0.960	0.959	0.987	0.974	0.960	0.957	0.958	0.993	
		ERGAS ↓ [105]	4.019	4.185	3.190	3.193	3.094	3.005	3.205	3.269	3.269	2.787	
		SAM ↓ [121]	0.0789	0.0822	0.0625	0.0626	0.0606	0.0589	0.0626	0.0628	0.0640	0.0556	
		UIQI ↑ [113]	0.3876	0.3817	0.5118	0.5036	0.5176	0.5424	0.5031	0.4874	0.4901	0.6169	
		sCC ↑ [137]	0.8681	0.8570	0.9210	0.9207	0.9253	0.9292	0.9195	0.9198	0.9157	0.9450	
		EPI ↑ [84]	0.3442	0.3690	0.3582	0.3567	0.4048	0.4253	0.3493	0.3844	0.3978	0.4750	
		PSI ↑ [30]	0.1806	0.1842	0.1856	0.1854	0.2401	0.1827	0.1766	0.1798	0.1849	0.2353	

‘↑’ means the higher the value, the better and ‘↓’ denotes that the smaller the value is, the better. Bold: indicates the best results.

Table 4.6: Performance evaluation of different methods for test images of LISS-III using $\times 2$, $\times 3$ and $\times 4$ zooming.

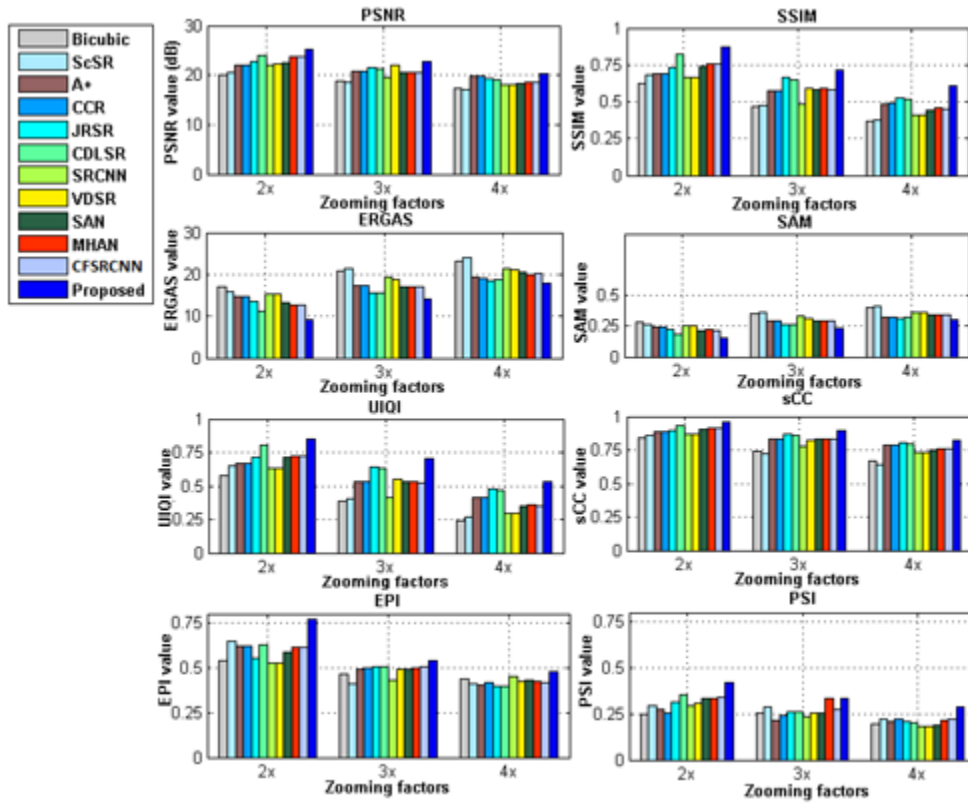
Zooming Factor	TestImage	Metric	LISS-III									
			Bicubic	SCSR	A+	CCR	JRSR	CDLSR	SAN	MHAN	CSRCNN	Proposed
$\times 2$	Test 7	PSNR (dB) \uparrow	30.27	30.67	31.43	31.42	32.87	32.24	31.75	32.14	32.04	34.40
		SSIM \uparrow	0.900	0.915	0.934	0.933	0.965	0.954	0.947	0.948	0.948	0.984
		ERGAS \downarrow [105]	3.632	3.467	3.182	3.181	2.692	2.896	3.070	2.941	2.971	2.437
		SAM \downarrow [121]	0.0710	0.0678	0.0622	0.0622	0.0526	0.0566	0.0592	0.0575	0.0579	0.0476
		UIQI \uparrow [113]	0.6465	0.6863	0.7109	0.7090	0.8002	0.7691	0.7699	0.7712	0.7698	0.8405
		sCC \uparrow [137]	0.9314	0.9376	0.9487	0.9486	0.9624	0.9564	0.9524	0.9554	0.9545	0.9770
		EPI \uparrow [84]	0.4781	0.4942	0.5595	0.5569	0.5240	0.4934	0.4911	0.5015	0.5030	0.5831
		PSI \uparrow [30]	0.2358	0.2510	0.2556	0.2502	0.3280	0.2905	0.2691	0.2790	0.2798	0.3592
		PSNR (dB) \uparrow	28.34	28.14	30.18	30.20	30.65	30.48	29.79	29.84	29.83	31.84
		SSIM \uparrow	0.800	0.799	0.887	0.888	0.909	0.902	0.878	0.876	0.873	0.929
$\times 3$	Test 7	ERGAS \downarrow [105]	4.533	4.639	3.662	3.659	3.478	3.545	3.835	3.811	3.812	3.301
		SAM \downarrow [121]	0.0887	0.0908	0.0716	0.0715	0.0680	0.0693	0.0744	0.0743	0.0742	0.0655
		UIQI \uparrow [113]	0.4802	0.4905	0.6124	0.6116	0.6650	0.6498	0.6232	0.6141	0.6099	0.6974
		sCC \uparrow [137]	0.8898	0.8843	0.9311	0.9312	0.9366	0.9340	0.9227	0.9230	0.9234	0.9498
		EPI \uparrow [84]	0.3075	0.3350	0.3539	0.3573	0.4029	0.4080	0.3812	0.3847	0.3846	0.4326
		PSI \uparrow [30]	0.2095	0.2208	0.2251	0.2189	0.2229	0.2186	0.1990	0.2069	0.2109	0.2941
		PSNR (dB) \uparrow	27.13	26.88	29.31	29.32	29.65	29.39	28.32	28.59	28.44	30.85
		SSIM \uparrow	0.714	0.713	0.841	0.841	0.864	0.855	0.799	0.807	0.804	0.888
		ERGAS \downarrow [105]	5.213	5.366	4.051	4.047	3.895	4.015	4.531	4.395	4.461	3.711
		SAM \downarrow [121]	0.1021	0.1051	0.0792	0.0792	0.0762	0.0785	0.0872	0.0858	0.0872	0.0735
$\times 4$	Test 7	UIQI \uparrow [113]	0.3501	0.3664	0.5227	0.5226	0.5722	0.5592	0.4715	0.4815	0.4780	0.6210
		sCC \uparrow [137]	0.8510	0.8425	0.9144	0.9146	0.9197	0.9151	0.8924	0.8958	0.8912	0.9334
		EPI \uparrow [84]	0.3032	0.3211	0.3821	0.3832	0.3700	0.3429	0.3327	0.3509	0.3340	0.3952
		PSI \uparrow [30]	0.1488	0.1467	0.1520	0.1523	0.1873	0.2504	0.1567	0.1568	0.1552	0.2489
		PSNR (dB) \uparrow	30.66	31.07	31.86	31.86	33.37	32.70	31.94	32.48	32.37	34.93
		SSIM \uparrow	0.971	0.975	0.988	0.988	0.996	0.995	0.985	0.986	0.985	0.998
		ERGAS \downarrow [105]	4.282	4.081	3.725	3.721	3.098	3.357	3.665	3.482	3.509	2.824
		SAM \downarrow [121]	0.0816	0.0778	0.0709	0.0708	0.0590	0.0640	0.0690	0.0663	0.0667	0.0556
		UIQI \uparrow [113]	0.6137	0.6573	0.6789	0.6761	0.7777	0.7416	0.7267	0.7288	0.7264	0.8125
		sCC \uparrow [137]	0.9547	0.9589	0.9662	0.9662	0.9762	0.9720	0.9680	0.9702	0.9698	0.9889
$\times 2$	Test 8	EPI \uparrow [84]	0.4066	0.4222	0.4558	0.4512	0.5190	0.4793	0.4739	0.4873	0.4882	0.5709
		PSI \uparrow [30]	0.1788	0.1967	0.1866	0.1923	0.1873	0.2504	0.1867	0.1868	0.1852	0.2489
		PSNR (dB) \uparrow	28.71	28.49	30.61	30.61	31.03	30.85	30.05	30.07	30.11	32.23
		SSIM \uparrow	0.927	0.928	0.977	0.977	0.988	0.985	0.967	0.965	0.963	0.989
		ERGAS \downarrow [105]	5.379	5.514	4.306	4.304	4.087	4.171	4.563	4.542	4.534	4.009
		SAM \downarrow [121]	0.1026	0.1052	0.0821	0.0820	0.0779	0.0795	0.0866	0.0864	0.0862	0.0754
		UIQI \uparrow [113]	0.4450	0.4561	0.5696	0.5680	0.6208	0.6033	0.5641	0.5541	0.5489	0.6619
		sCC \uparrow [137]	0.9272	0.9233	0.9543	0.9544	0.9582	0.9564	0.9485	0.9489	0.9491	0.9701
		EPI \uparrow [84]	0.3297	0.3439	0.4363	0.4425	0.3963	0.3876	0.3673	0.3731	0.3736	0.4676
		PSI \uparrow [30]	0.2120	0.2100	0.2120	0.2148	0.2145	0.2105	0.1945	0.2151	0.2178	0.2989
$\times 4$	Test 8	PSNR (dB) \uparrow	27.49	27.22	29.71	29.71	30.02	29.79	28.52	28.83	28.66	31.25
		SSIM \uparrow	0.893	0.904	0.959	0.959	0.971	0.985	0.927	0.929	0.929	0.992
		ERGAS \downarrow [105]	6.194	6.393	4.779	4.778	4.597	4.737	5.431	5.254	5.320	4.490
		SAM \downarrow [121]	0.1183	0.1221	0.0911	0.0911	0.0877	0.0903	0.1023	0.1000	0.1014	0.0842
		UIQI \uparrow [113]	0.3232	0.3412	0.4748	0.4742	0.5207	0.5176	0.4139	0.4192	0.4182	0.5671
		sCC \uparrow [137]	0.9022	0.8959	0.9433	0.9434	0.9469	0.9440	0.9280	0.9311	0.9289	0.9495
		EPI \uparrow [84]	0.3089	0.3226	0.3387	0.3346	0.3428	0.3351	0.3135	0.3240	0.3256	0.3689
		PSI \uparrow [30]	0.1807	0.1735	0.1765	0.1848	0.1776	0.2441	0.1688	0.1817	0.1811	0.2457

' \uparrow ' means the higher the value, the better and ' \downarrow ' denotes that the smaller the value is, the better. **Bold**: indicates the best results.

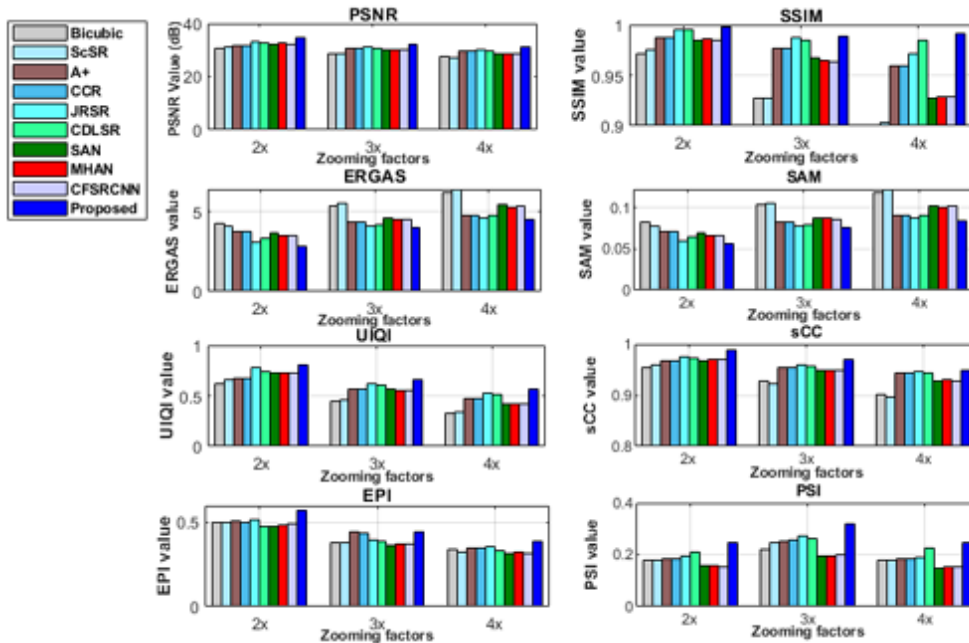
imposed by the image degradation model on the input images before feeding to the networks for the training. It will be interesting to see if the existing DL models can be modified to counter the degradation component as well in our subsequent works. The proposed method is also compared with the recent generative adversarial network (GAN)-based SR method i.e best-buddy GAN (Beby-GAN) [59], and reveals that although the GAN-based method achieves excellent perceptual results, the PSNR is 2–3 dB lower than the proposed method. The proposed method produces the best visual results supported by higher PSNR and better edge preservation. In a subsequent experiment, some other sparse representation-based approaches and the recent DL-based SR methods, i.e. SAN, CFSRCNN and MHAN are compared with the proposed method on LISS-IV and LISS-III datasets. Averaged results for LISS-III and LISS-IV computed over all the band images for each test image are shown in Tables 4.6 and 4.5. In terms of PSNR, on an average, the proposed method outperforms others in the ranges of 1.9–6.5 dB, 1.4–3.7 dB, and 1.5–4.2 dB for $\times 2$, $\times 3$, and $\times 4$ upscaling, respectively on LISS-IV images. In case of LISS-III, the proposed method achieves the highest PSNR compared to other methods by an improvement of 1.5–4.1 dB, 1.1–3.5 dB, and 1.2–3.7 dB, respectively, for $\times 2$, $\times 3$, and $\times 4$ zooming. Furthermore, the quantitative measures on ‘Test-4’ and ‘Test-8’ images are shown graphically in Fig. 4.7. The above detailed analysis shows that the proposed method is able to maintain the reconstruction quality as well as retain edges and structural features more effectively.

4.4.4 SR on real MS images

In order to validate the robustness of the proposed method, a comparison experiment is also carried out on real-world MS remote sensing images. Unlike the previous experiments, real-world LR remote sensing image is not subjected to synthetic blurring and downsampling. This LR image is fed directly to the proposed method and other SR methods for $2\times$ and $4\times$ zooming. We use two non-reference-based quantitative metrics, i.e., NIQE [67] and entropy (EN), to evaluate the outputs, as the ground-truth HR image is not available in this case. Ideally, a lower score of NIQE and



(a) Test-4 of PatternNet



(b) Test-8 of LISS-III

Figure 4.7: Performance evaluation of various methods on Test-4 of PatternNet and Test-8 of AID for different zooming factors.

higher score of EN indicate a better reconstruction result. Here, we have conducted an experiment on a LISS-IV real LR image of size 256×256 for $2 \times$ and $4 \times$ zooming, as shown in Fig. 4.8, where NIQE and EN values are given along with the visual results. We have compared the proposed method with sparse-representations-based and one of the best performing DL method i.e MHAN. From visual results and quantitative metrics shown in Fig. 4.8, it is found that the proposed method obtains lower NIQE and higher EN with better visual results.














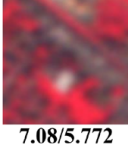

Input	ScSR	A+	CCR	JRSR	CDLSR	MHAN	Proposed
							
	5.89/5.701	5.77/5.768	5.71/5.770	5.22/5.783	5.64/5.762	5.72/5.779	5.16/5.801
ROI							
NIQE/EN	7.34/5.698	7.12/5.754	7.09/5.769	7.01/5.772	7.05/5.759	7.08/5.772	6.50/5.791

Figure 4.8: SR results of LISS-IV images without blurring for different methods at different scales.

4.4.5 Convergence test and Complexity analysis

An experiment on the proposed dictionary learning is performed by observing the PSNR of the reconstructed image to determine convergence. Fig. 4.9 presents the convergence plot for ‘Test-5’ at $2 \times$ zooming, over 30 iterations. The optimal stopping point is identified at the 20^{th} iteration, beyond which PSNR improvements level off, suggesting that PSNR improvements remain uniform beyond this point, suggesting that the proposed dictionary learning method converges effectively.

The computational complexity of the proposed dictionary learning method is significantly influenced by the K-SVD algorithm, which is central to the learning process. The K-SVD step iteratively updates dictionary atoms and performs sparse coding of all patches. The complexity of this step: $\mathcal{O}(T \times (n \times t \times k \times r^2 + k \times (m \times r^2 + r^3)))$, where T denotes the number of iterations, n the number of patches, t the sparsity level, k the number of dictionary atoms, r the patch size, m the number

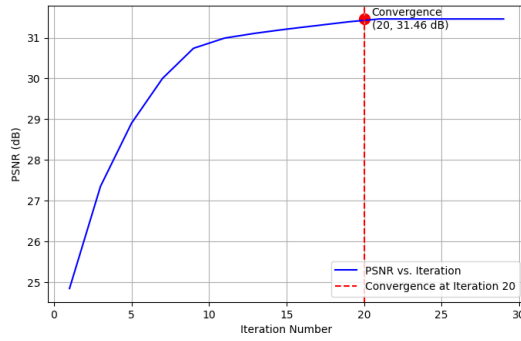


Figure 4.9: Convergence analysis of the proposed method on ‘Test-5’ for $2\times$ zooming: PSNR vs number of iterations.

of feature maps. In this proposed method, two dual dictionary learning processes are implemented using KSVD: keypoints-based and non-keypoints-based coupled dictionary learning. Each process operates independently within the same overall algorithmic framework. Given this dual-dictionary approach, the computational loads of the sparse coding and dictionary update steps from each dictionary are cumulative. Consequently, the total time complexity for the method is $\mathcal{O}(2 \times T \times (n \times t \times k \times r^2 + 2 \times k \times (m \times r^2 + r^3)))$, reflecting the increased computational demand due to constructing two distinct dictionary learning processes simultaneously.

4.4.6 Parameters empirical study

- (i) **The effect of patch size and numbers of overlapping pixels:** The performance of the proposed method depends on the patch size and number of overlapping pixels. Different patch size with varying number of overlapping pixels are applied to LISS-IV Test-5 image for zooming factor 2 to check their impact on the proposed method as shown in Fig. 4.10a. The PSNR values for overlapping of 2, 3, and 4 obtained by patch size 5×5 are 32.46, 32.48, and 32.50, respectively. When the patch size is 7×7 and the number of pixel overlappings ranges from 4 to 6, the PSNR results are 32.59, 32.60, and 32.62, respectively. Similarly, The PSNR results for patch size 9×9 with number of overlapping pixels 6 to 8 are 32.58, 32.59 and 32.60, respectively. The results in Fig. 4.10a indicate that the highest PSNR for the proposed method is achieved when the patch size and number of overlapping pixel are 7×7 and

6, respectively. Therefore, these values are selected for the proposed method.

- (ii) **The effect of λ :** Fig. 4.10b shows the PSNR obtained by the proposed method on LISS-IV Test-5 image for zooming factor 2 by varying the λ value ranges from 1.2 to 1.8. It is observed from Fig. 4.10b that the PSNR value reaches its maximum when λ value is 0.15.
- (iii) **Trade-off parameters μ_1 and μ_2 :** The selection of optimum values for μ_1 and μ_2 in Eq. 4.11 is essential as it indicates the relative importance of NLTV- and key-point sparsity priors. Here, we consider LISS-IV Test-5 image as examples to demonstrate how μ_1 and μ_2 impact on the performance of the proposed method in zooming factor 2. The PSNR surface plot for Test-5 are shown in Fig. 4.10c, where μ_2 values range from 0.001 to 0.007 and the ratio of μ_1 to μ_2 varies from 0 to 0.6. It is observed that the highest PSNR values are achieved when the value of μ_2 is 0.005 and μ_1 to μ_2 ratio is between 0.2 – 0.4. Therefore, the final optimal values of μ_1 and μ_2 for the proposed method are selected as 0.0015 and 0.005, respectively.

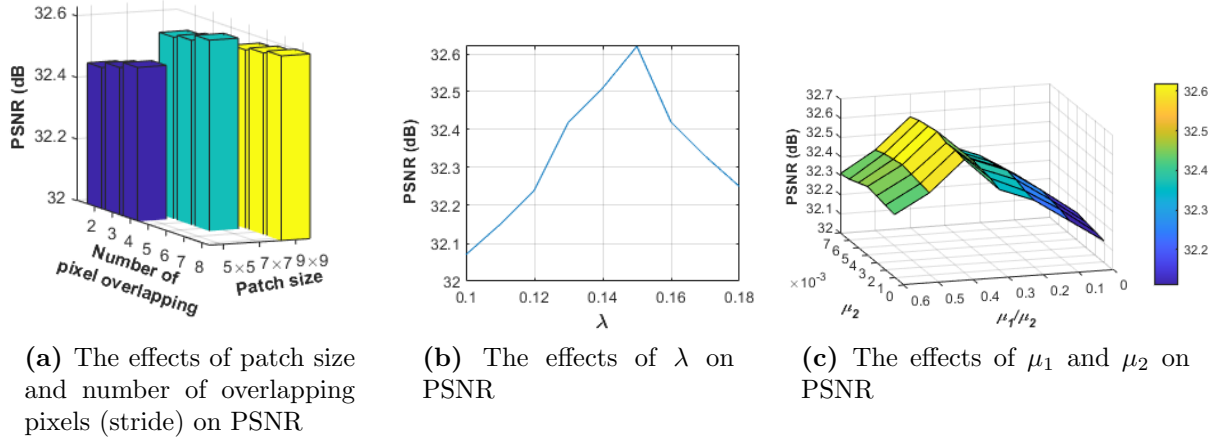


Figure 4.10: PSNR vs. algorithm parameters.

4.4.7 Scalability Study

Experiments are also performed to determine the efficacy and scalability of the proposed GPU implementation with respect to different factors, such as dictionary

and input image sizes. In order to train both the D_h and D_ℓ , simultaneously, fixed size sample patches of around 100,000 are used. It is observed that there is no difference in CPU execution time for different zooming factors. Fig. 4.11a shows the impact of changing the dictionary size on the execution time of the proposed CPU and CUDA-GPU-based dictionary learning. The speed-up ranges from 5.4 to 7 as the dictionary size increases from 256 to 1024 on GPU. Because DL-based SR models need approximately 7-12 hours to train, the training time taken for comparison are primarily shown for sparse dictionary learning-based approaches against the proposed method in Fig. 4.11b. The GPU implementation takes only a few seconds to train the dictionary. It gives on an average $150\times$ acceleration over other methods for a dictionary size of 512.

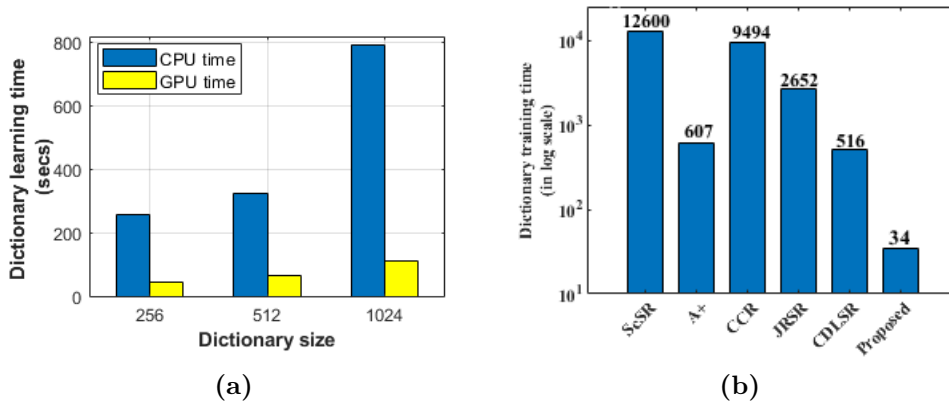


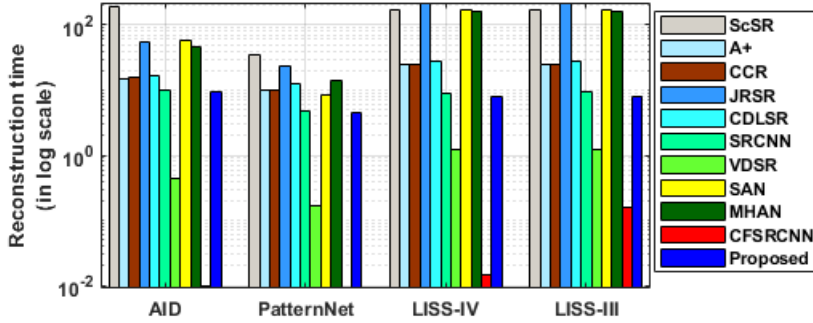
Figure 4.11: (a) CPU VS. GPU dictionary training time for different dictionary sizes on LISS-IV dataset (b) Dictionary training time comparison of different methods for 512 dictionary size (dictionary training time in secs. are shown above the bars).

The runtime of CPU and GPU reconstructions for $2\times$, $3\times$, and $4\times$ zooming factors are shown in Table 4.7 for LISS-IV images. The speed-up factors of GPU reconstruction increase gradually from 60 to 122 times and 116 to 179 times for $2\times$ and $3\times$ zooming factors, respectively when compared to their CPU counterparts as image size increases from 256×256 to 1024×1024 . Similarly, reconstruction time is reduced by 122 to 213 times while using the proposed method for zooming factor 4, which only takes 13-46 seconds. Average reconstruction times taken by test images of different datasets for different methods are plotted as shown in Fig. 4.12 for $2\times$ zooming. It can be observed that reconstruction time of the proposed method is faster than other dictionary learning-based methods. Despite some of the DL-based

Table 4.7: CPU vs. GPU reconstruction speed-up for different image sizes and different zooming factors: $\times 2$, $\times 3$ and $\times 4$.

Image size	CPU			GPU			Speed-up		
	$\times 2$	$\times 3$	$\times 4$	$\times 2$	$\times 3$	$\times 4$	$\times 2$	$\times 3$	$\times 4$
256×256	312.23	703.45	1240.56	5.2	7.5	10.10	60.04	93.79	122.82
512×512	1250.64	2451.34	3939.87	10.7	16.25	22.00	116.88	150.85	179.08
1024×1024	3450.41	6023.52	9845.25	18.50	30.50	46.03	186.50	197.49	213.88

SR methods such as VDSR, CFSRCNN are faster than the proposed method in terms of reconstruction time, the reconstruction quality of these methods is noticeably poor. Additionally, the time taken by the computationally exhaustive parts of the proposed algorithm for 256×256 image size with zooming factor 2 on CPU and GPU are shown in Table 4.8. It is visible that the CUDA-based implementations of individual sequential counterparts can reduce the computational time to a greater extent.

**Figure 4.12:** Comparison of average reconstruction time of different methods across different datasets.**Table 4.8:** Performance of CUDA-Implementation over CPU for computationally exhaustive operations in the proposed algorithm.

Operation	CPU time (secs.)	GPU time (secs.)	Speed-up
K-SVD	272.21	48.32	5.63
Patch-based Reconstruction	128.24	2.31	55.51
Keypoint-driven patch-based reconstruction	10.70	0.17	62.90
ADMM	153.12	3.05	50.20

Analysis of reconstruction time for real RS image is very essential since these images are typically very large in size. It only takes a few minutes (~ 450 secs.) to process real RS images (up to 3000×1500 image) for the zooming factor 4 using the proposed CUDA-GPU SR method. Due to memory constraints, CPU-based

sequential implementations are unable to process real RS images.

4.5 Conclusion

In this chapter, a highly parallelized SISR framework accelerated with CUDA-GPU implementation of edge preserving coupled dictionaries- SIFT-based keypoints driven and non-keypoints patch-based dictionaries, and sparse representations is presented. On the basis of both SIFT keypoints-guided patch sparsity and NLTV-based patch sparsity, a joint reconstruction model is developed to preserve high frequency features (edges). Visual results and objective criteria clearly demonstrate that it outperforms the state-of-the-art sparse representation- and DL-based SR methods. CUDA-GPU implementation demonstrates significant speed-up as compared to sequential implementations and holds great potential for RS applications.

The main limitation of the proposed work is that sparse representations-based SISR method relies on the quality of the hand-crafted features extracted from the LR image and sparsity for effective training of the dictionaries required for reconstruction of the corresponding HR image. However, these requirements may not always hold true in practice for the LR remote sensing image. The benefits of automatic feature extraction in DL-based SISR methods include more data-driven and flexible solutions, the ability to handle complex mappings between LR and HR images, and the ability to interpret the complex structure of the image without requiring manual feature selection. These advantages make DL-based SISR methods a powerful tool for image SR, especially in applications of RS. However, we have seen that existing DL-based SISR networks cannot alone deal with the blurriness of LR remote sensing images. In the future, we will propose an end-to-end dual-branch DL network consisting of independent deblurring and SR modules. This network will be able to restore LR remote sensing images degraded with Gaussian blur to produce sharp and clear HR images.