CLOUD REMOVAL IN HYPERSPECTRAL IMAGES

B.Tech Thesis Project
by

ANIRBAN MITRA
14EE10004

Under the supervision of
Dr. Rajiv Ranjan Sahay
(Department of Electrical Engineering,
IIT Kharagpur)

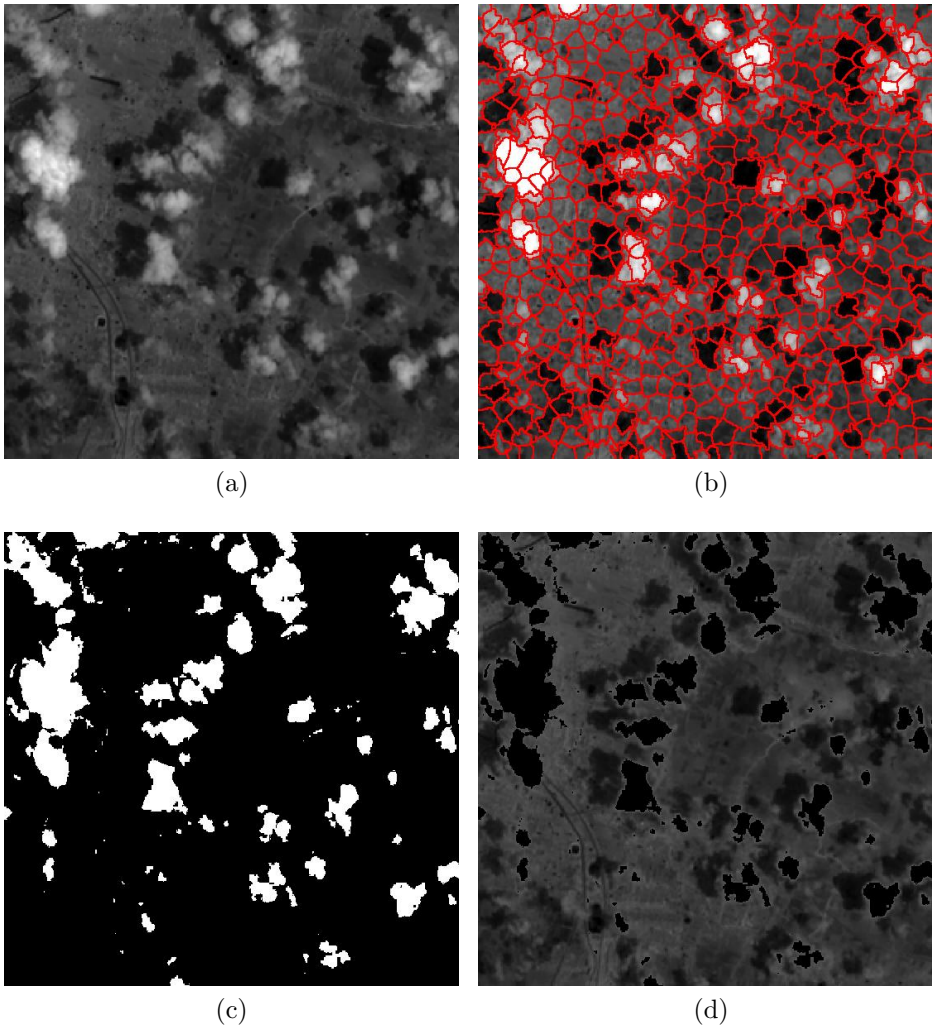
Department of Electrical Engineering
Indian Institute of Technology Kharagpur
Kharagpur - 721302, India

Abstract

Cloud Coverage is a major problem in the optical remote sensing imagery. Hence automated cloud detection and cloud removal is an important step needed in remote sensing image processing. After detection of the cloud mask, successful reconstruction of the occluded region is a major challenge. For successful in-painting, there exists various algorithms which are based on texture synthesis, PDE based in-painting, Hybrid in-painting etc. The algorithm explored to in-paint the image is based on Sparse Representation Modelling[1]. There has been a growing use in the sparse representation of signals in fields of in-painting, feature extraction, de-noising etc as described in [2]. By using an over-complete dictionary that contain prototype signal atoms, the objective signals can be described as the linear sparse combination of these target atoms. For inpainting gradient descent algorithm is used using Gaussian Markov Random Field as a prior. This prior would help in diffusion based inpainting of the image. Further I have explored Total Variation(TV) inpainting which uses the Split Bregman method to inpaint images as described in [3].

Detection Of Clouds

For Cloud Detection, I used super-pixel segmentation which divides the image into N cluster based on similarity of pixels. It uses simple linear iterative clustering algorithm to group pixels into regions with similar values. The mask generated is passed to the active contour algorithm which segments the image in a mask which contains cloudy and non-cloudy regions.



(a) (b)

(c) (d)

Figure 1: (a) Original Cloud Image, (b) Super Pixel Image, (c) Mask Formed, (d) Occluded Portion in Original Image.

Reconstruction Of the Occluded Portions

For reconstruction Sparse Representation Modelling is used. Sparse Representation theory says that sparse signals can be exactly reconstructed from small number of elementary atoms. Sparse Dictionary Learning is a representation learning which aims at finding a sparse representation of the input data in the form of a linear combination of basic elements. These elements are called atoms and they compose a dictionary. Given a set of training examples we aim to find the dictionary that leads to best representation of each member under strict sparsity constraints. Thus given an overcomplete dictionary, a target signal y can be represented as a sparse linear combination of dictionary atoms

$$Y = DX \tag{1}$$

where X is a sparse coded vector, Y the set of original signals and D, the overcomplete dictionary. Figure.5 shows the relation between Y, D and X.

For finding the sparse coded vector using OMP under the constraints,

$$\min_{x} \|Y - DX\|_2^2 \quad subject \quad to \quad \|x\|_0 \leq T \tag{2}$$

where $||.||_0$ represents the $L-0$ norm(Quasi norm) or the number of non-zero values in the signal. Here It is maximum sparsity taken. A similar objective can alternately be met by considering the MSE being less than some threshold value.

$$\min_{D,X} \sum_{i} \|x\|_0 \quad subject \quad to \quad \|Y - DX\|_2^2 \leq \epsilon \tag{3}$$

We use algorithms which greedily approximates solutions like Matching Pursuit and Orthogonal Matching Pursuit to find the approximate sparse coded vector X. The basis pursuit algorithm is also a representative algorithm similar to OMP that solves the problem by replacing the L0 -norm with an $L1$ norm.

Orthogonal Matching Pursuit Algorithm for Finding Sparse Coded Vector

Orthogonal Matching Pursuit (OMP) is a representation algorithm for finding the sparse coded vector. It involves finding the best matching projection of multidimensional data onto the span of a dictionary. As it is a greedy algorithm, on every step it selects the atom with the highest correlation with the existing method. On selecting the atom, the signal is projected orthogonally to the selected atoms and the residual is re-calculated. This method is repeated until stopping criterion is met. The steps given below describes the algorithm.

Algorithm 1 ORTHOGONAL MATCHING PURSUIT

1: Input: Dictionary D, signal $\underline{x}$, target sparsity K or target error ϵ
2: Output: Sparse Representation $\underline{\gamma}$ such that $\underline{x} \approx D\underline{\gamma}$
3: Init: Set $I := ()$, $\underline{r} := \underline{x}$, $\underline{\gamma} := \underline{0}$
4: **while** stopping criterion not met **do**
5: $\quad \hat{k} := \text{argmax}_k \left| \underline{d_k^T r} \right|$
6: $\quad I := (I, \hat{k})$
7: $\quad \underline{\gamma_I} := (D_I)^+ \underline{x}$
8: $\quad \underline{r} := \underline{x} - D_I \underline{\gamma_I}$
9: **end while**

KSVD Algorithm

For finding the optimal Dictionary, K-SVD algorithm is used. K-SVD, a dictionary learning algorithm is used for creating the optimal dictionary for sparse representations, via the singular value decomposition approach [1]. SVD decomposition is used to maximize the similarity of the existing dictionary. K-SVD is similar to the k-means clustering method as it works by iteratively alternating between sparse coding the target signal on basis of the current dictionary, and simultaneously updating the atoms to fit the signal properly.

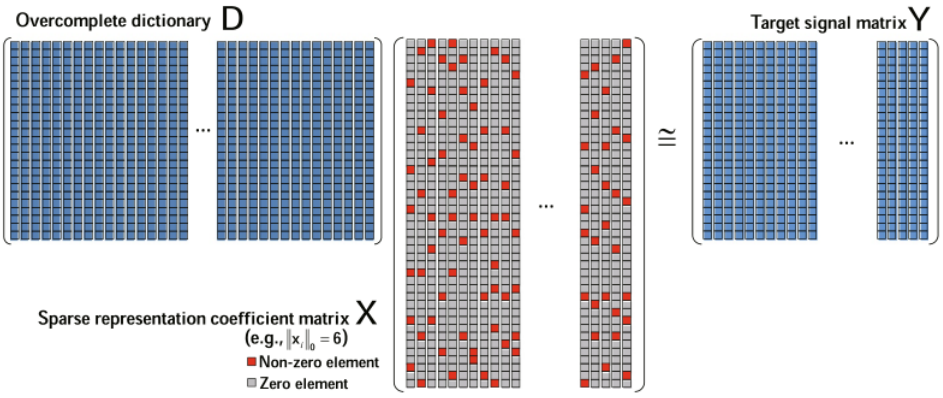


Figure 2: Sparse Signal Representation of Y

The update of atoms in the dictionary is combined with the simultaneous update of the sparse coded vector to make convergence faster. The K-SVD algorithm is flexible in the sense that it can work with almost any matching pursuit algorithms.

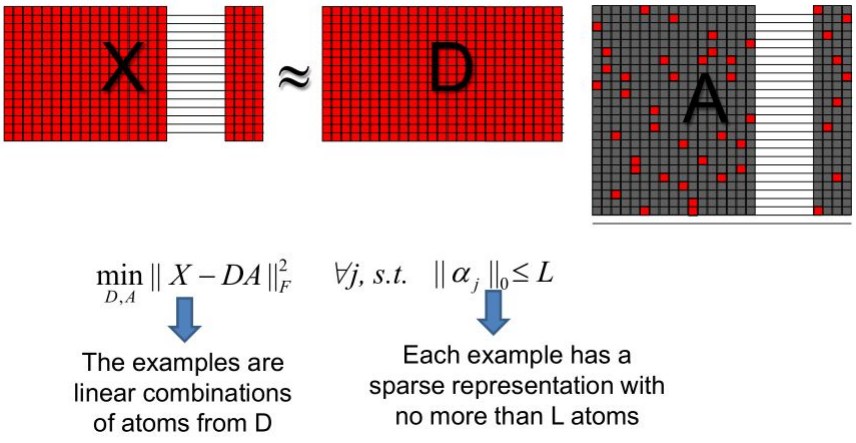


Figure 3: Objective Function For KSVD

The dictionary updated in an iterative manner as depicted in the flowchart:

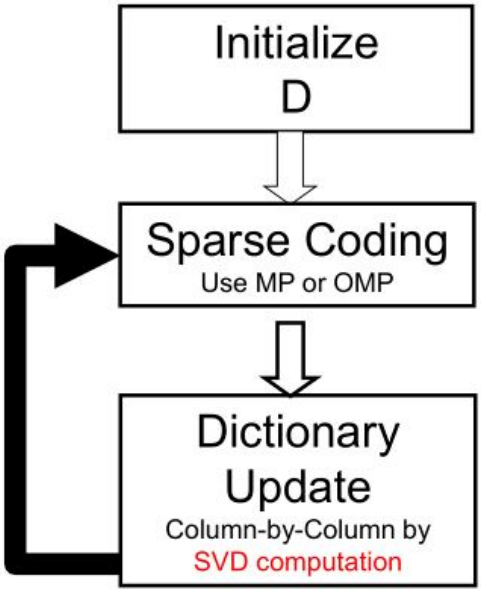


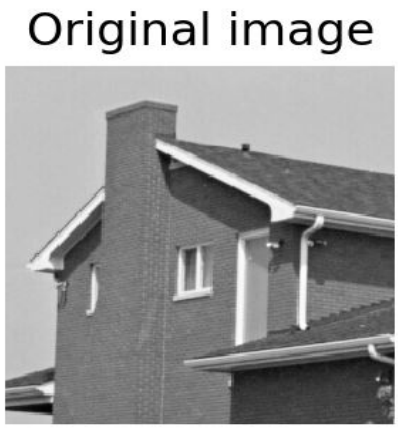
Figure 4: Objective Function For KSVD

Results in Denoising of Images

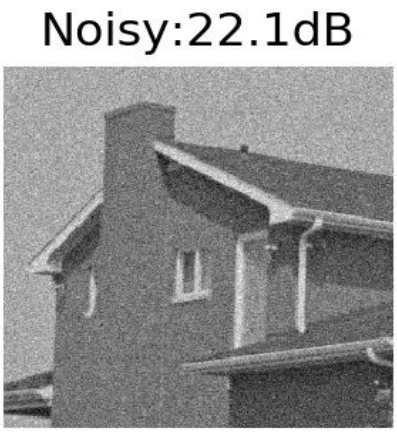
This KSVD algorithm was applied for the de-noising of images. The results obtained were in par with other de-noising algorithms. The noise added was Gaussian noise.

This same process was repeated for a RGB image by taking the concatenation of different bands as the vector but the results obtained were not that good.

This method was repeated taking a patch size of $8 * 8$ throughout the image. On taking distinct patches the overall results were not up to the mark, so overlapping patches were taken to train the Dictionary well.



(a)



(b)

(c)

Figure 5: (a) Original Image, (b) Noisy Image with Gaussian Noise added, (c) Denoised Image.

Thus we see that MSE denoising is not up to the mark.

Changing the Metric to SSIM

The original metric used in KSVD algorithm was the Mean Squared Error (MSE), as it is a popular metric used in most image processing algorithms. However, it is not that accurate as MSE cannot provide high visual quality in images. Thus it may not be appropriate to use MSE as quality measure in in-painting.

Another well-known metric measure is the Structural Similarity Index (SSIM), which is used for quality measure in Image Processing. The definition of the SSIM index is first extended to measure the similarity between signals via their statistical equivalents. The SSIM between two signals x and y depends on the sample means, sample variance and the cross-variance between x and y.

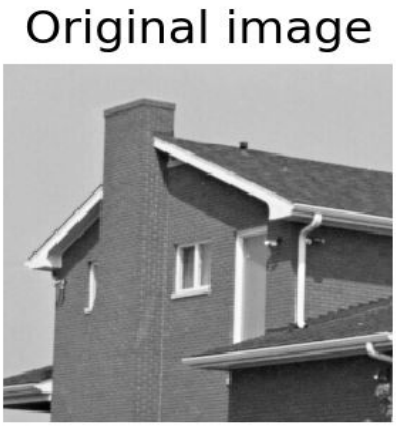
Thus the SSIM between the two signals is given by the equation:

$$SSIM(x,y) = \left(\frac{2\mu_x\mu_y + C_1}{\mu_x^2 + \mu_y^2 + C_1}\right)\left(\frac{2\sigma_{xy} + C_2}{\sigma_x^2 + \sigma_y^2 + C_2}\right) \tag{4}$$

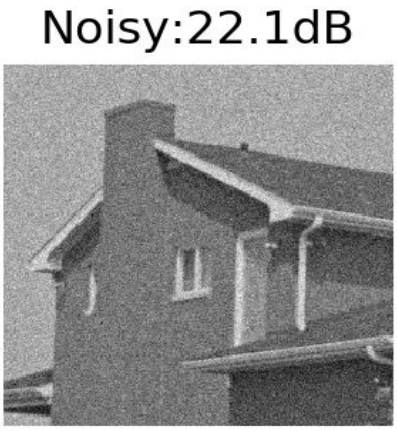
Thus instead of using MSE as a criterion in OMP for obtaining the sparse coded vector SSIM is used as a metric. Thus the objective function gets converted to

$$\max_{x_i} SSIM(y_i, Dx_i) \quad subject \quad to \quad \|x\|_0 \leq T \tag{5}$$

That is, we have to find a sparse coded vector such that the SSIM is maximum and also the sparsity does not exceed T. Thus SSIM metric will be the stopping criteria for the OMP algorithm. Thus on modifying the KSVD algorithm again using SSIM as a metric we apply it on de-noising of images [4].



(a)



(b)

(c)

Figure 6: (a) Original Image, (b) Noisy Image with Gaussian Noise added, (c) Denoised Image by SSIM 33.7dB.

SSIM Calculation

The SSIM metric is generally calculated in 2 different ways

- Using the dependance of means and standard deviation on pixel p
- By ommiting the dependence on pixel p and calculating the mean and standard deviation with a Gaussian filter with standard deviation σ_G

The expression of $SSIM$ can be written as:

$$SSIM(x,y) = \left(\frac{2\mu_x\mu_y + C_1}{\mu_x^2 + \mu_y^2 + C_1}\right)\left(\frac{2\sigma_{xy} + C_2}{\sigma_x^2 + \sigma_y^2 + C_2}\right) = l(p).cs(p) \tag{6}$$

For calculations via Gaussian Filter, given a patch P_x

$$\mu_x(p) = G_{\sigma_G} \circledast P_x \tag{7}$$

$$\sigma_x^2(p) = G_{\sigma_G} \circledast P_x^2 - \mu_x^2(p) \tag{8}$$

$$\sigma_{xy}(p) = G_{\sigma_G} \circledast (P_x.P_y) - \mu_x(p)\mu_y(p) \tag{9}$$

where, P_x is a patch centered at pixel p, '$\circledast$' denotes a convolution, '.' a point-wise multiplication. The values of $\mu_x(p)$ and $\sigma_y^2(p)$ can be computes similarly. The dimension of the filter is taken by the $6\sigma + 1$ principle so that it has a coverage over 95%.

Using Gradient Descent for De-Bluring

As the performance of SSIM does not come up to the mark, I checked whether it performs basic operations like deblurring in images, Thererfore a comparison was made between the performance of SSIM and MSE to see which one performs better.

Thus a comparison was based on 3 methods:

1. Gradient Descent using Mean Squared Loss as Objective Function.
2. Gradient Descent using SSIM with Gaussian Filter as Objective Function.
3. Gradient Descent using SSIM without Gaussian Filter as Objective Function.

The update equation in Gradient Descent is as:

$$\theta^{k+1} = \theta^k - \eta \circledast \nabla_\theta J(\theta) \tag{10}$$

where $J(\theta)$ is the objective function and η is the learning rate parameter that determines the size of steps we take to reach the minimum.

As we see from the below images that the weighted sum of MSE and SSIM performs the best in the above examples. This is because int MSE, the rate of convergence is really fast while on the other hand SSIM converges very slowly. SSIM is needed to reconstruct the structural components of the images. The graphs below show the performance of the three Objective Functions.

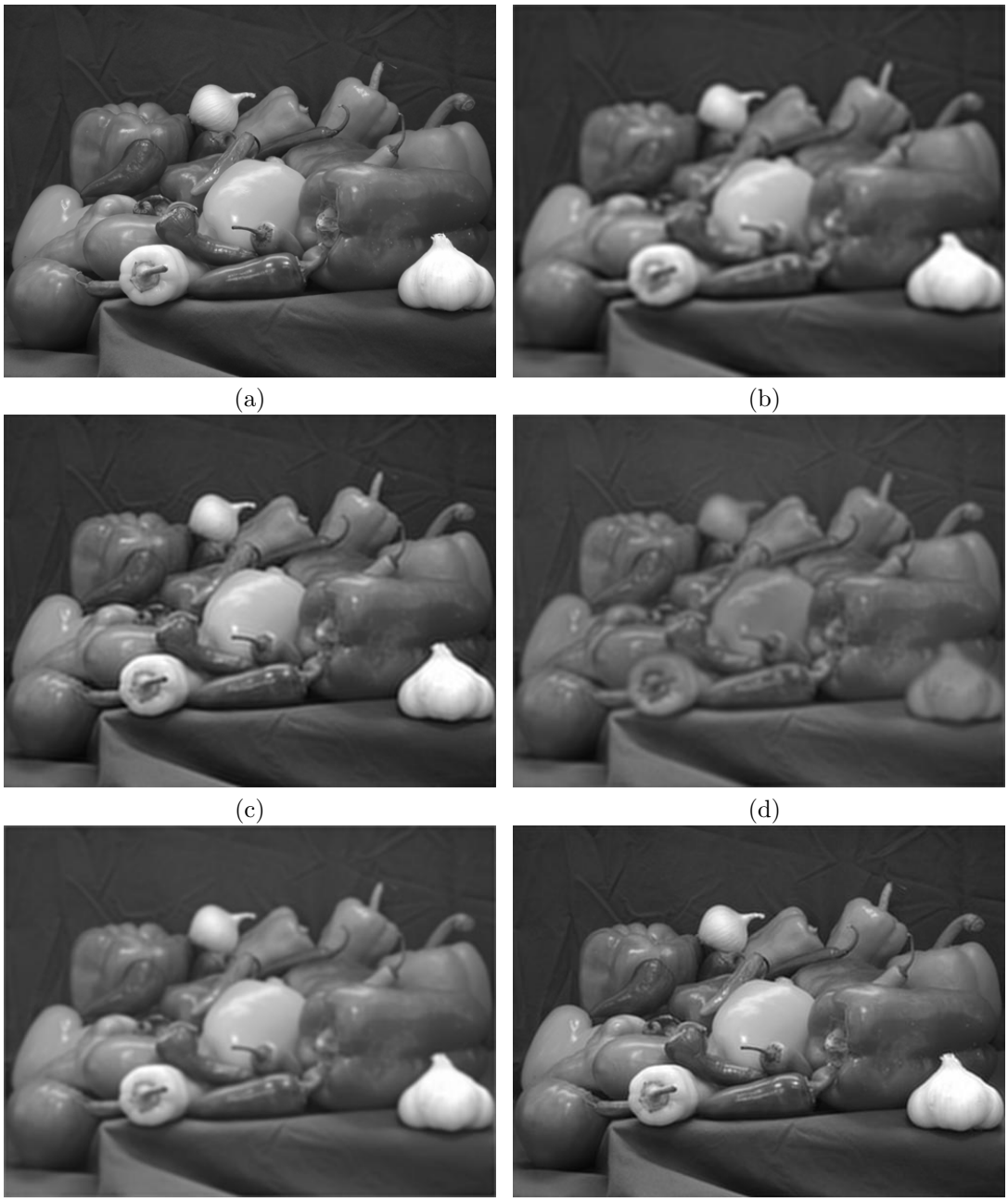


Figure 7: (a) Original Image Ground Truth, (b) Blurred Image, (c) Image Reconstructed via MSE, (d) Image Reconstructed via SSIM using Gaussian Filter, (e) Image Reconstructed via SSIM without using Gaussian Filter, (f) Image Reconstructed via weighted sum of MSE and SSIM.

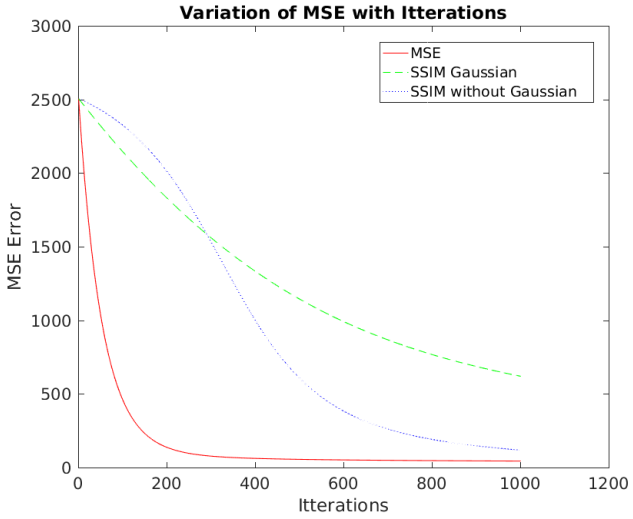


Figure 8: Variation Of MSE for 3 Objective Functions for 1000 itterations

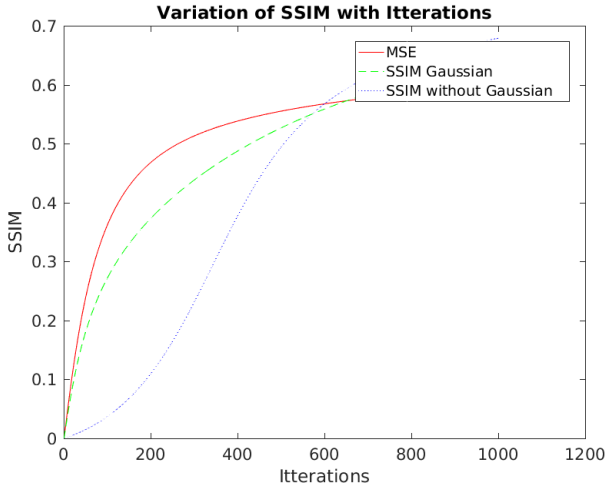


Figure 9: Variation Of SSIM for 3 Objective Functions for 1000 iterations

Using Gradient Descent for Inpainting

On applying of gradient descent for inpainting we use the GMRF model for diffusion inpainting. Diffusion inpainting makes use of the information in the neighbouring pixels to in-paint an unknown pixel. So the inpainting equation becomes.

$$Y = O. * X \tag{11}$$

where Y is our observation, X is our estimation and O is the mask. Thus in gradient descent the update equation becomes,

$$\theta^{k+1} = \theta^k - \eta \circledast (\nabla_\theta J(\theta) + lambda * prior) \tag{12}$$

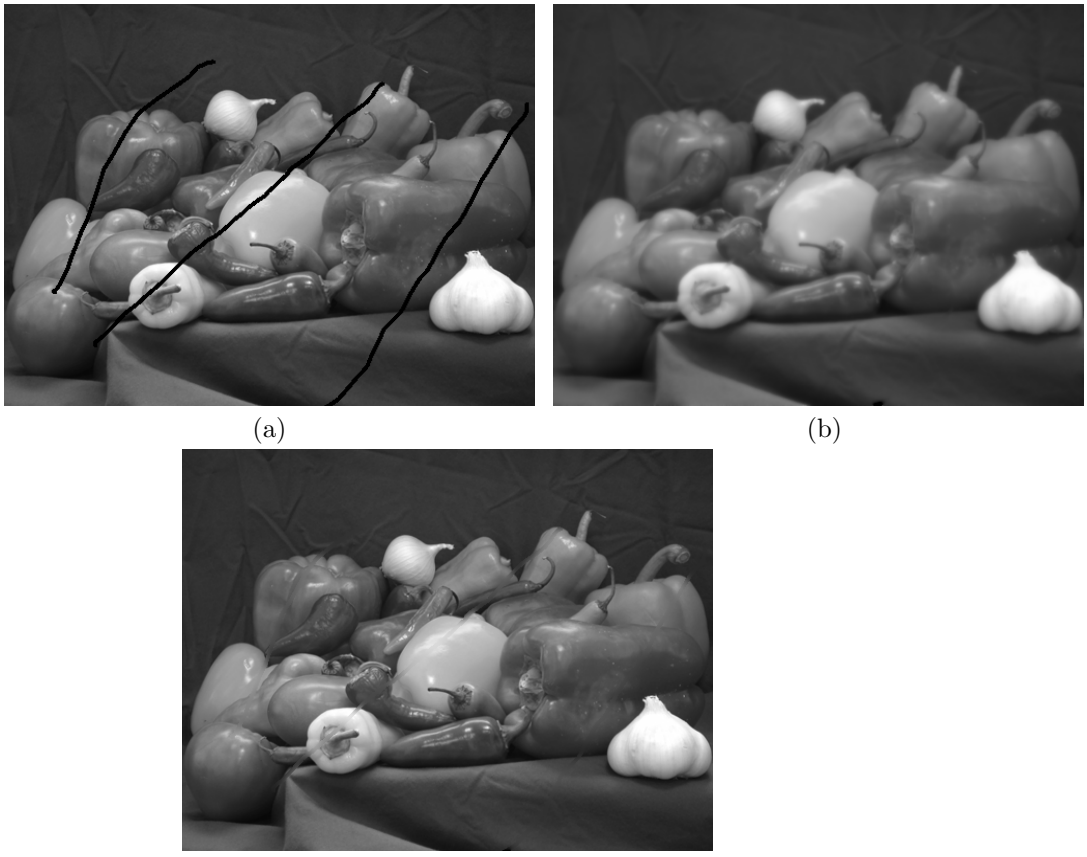
The GMRF model taken is the squared difference of the neighboiring pixels,

$$\begin{aligned} GMRF(i,j) = (X(i,j) - X(i-1,j))^2 + (X(i,j) - X(i+1,j))^2 + \\ (X(i,j) - X(i,j-1))^2 + (X(i,j) - X(i,j+1))^2 \end{aligned} \tag{13}$$

The derivative of the GMRF is calculated as:

$$\begin{aligned} \nabla GMRF(i,j) = 2 * ((X(i,j) - X(i-1,j)) + (X(i,j) - X(i+1,j)) + \\ (X(i,j) - X(i,j-1)) + (X(i,j) - X(i,j+1))) \end{aligned} \tag{14}$$

Results Obtained

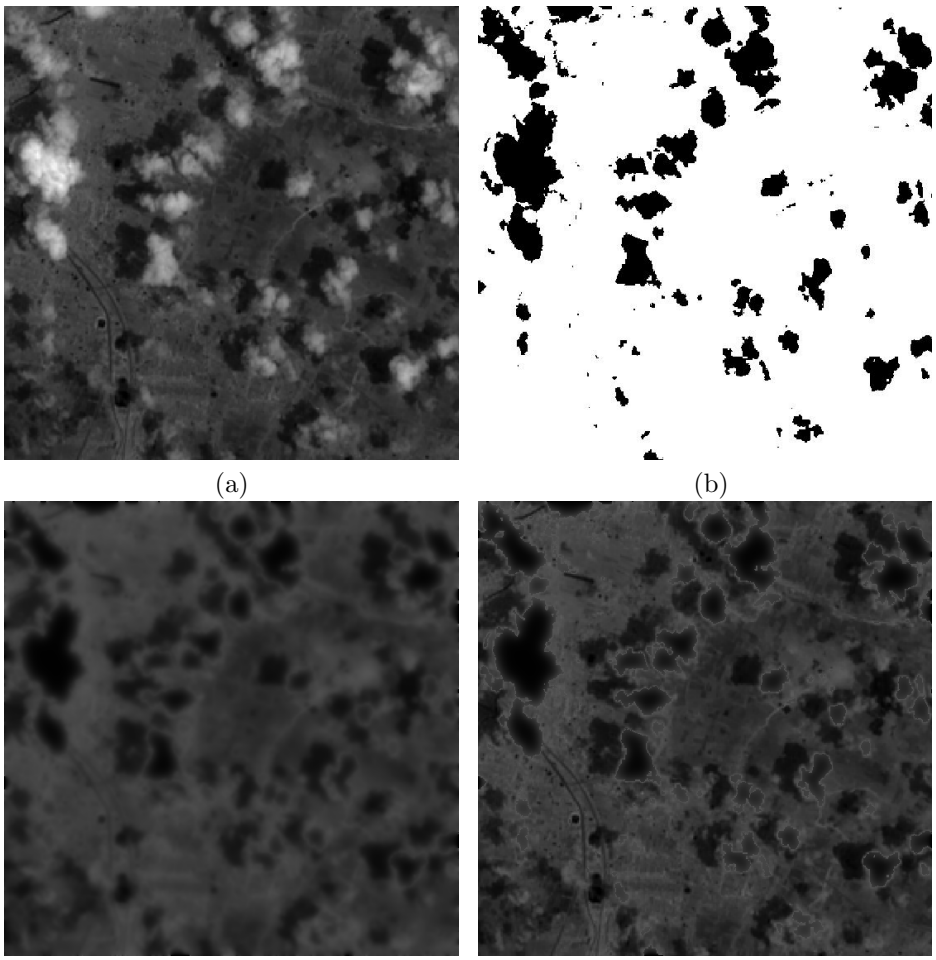


(a) (b)

(c)

Figure 10: (a) Observation Image which has to be In-Painted, (b) Estimated Image after Inpainting, (c) Output Image.

Results on Hyperspectral Images



(a) (b)

(c) (d)

Figure 11: (a) Cloud sample Image, (b) Cloud Mask, (c) Estimated Image, (d) Best Estimated Image.

In-Painting Using Split Bregman Itteration

Until now I have used GMRF as the prior, but research shows that TV(Total Variation) as a prior performs well. TV denoising is considered to be one of the best denoising models, but also one of the hardest to compute. However Bregman itteration technique can be used to solve this problem extremley simply and efficiently. Total Variation can be defined as

For an image the TV can be of two types:

- Isotropic Total Variation

$$V(y) = \sum_{i,j} \sqrt{\left|y_{i+1,j} - y_{i,j}\right|^2 + \left|y_{i,j+1} - y_{i,j}\right|^2} \tag{15}$$

  which for image can be calculated as:

$$\min_u \sum_i \sqrt{(\nabla_x u)_i^2 + (\nabla_y u)_i^2} \tag{16}$$

- Anisotropic Total Variation

$$V(y) = \sum_{i,j} \left|y_{i+1,j} - y_{i,j}\right| + \left|y_{i,j+1} - y_{i,j}\right| \tag{17}$$

$$\min_u |\nabla_x u| + \left|\nabla_y u\right| \tag{18}$$

The Split Bregman Expression can be shown as:

$$\min_{d_x, d_y, u} |d_x| + \left|d_y\right| + \frac{\mu}{2}\|u - O.X\|_2^2 + \frac{\lambda}{2}\left\|d_x - \nabla_x u - b_x^k\right\|_2^2 + \frac{\lambda}{2}\left\|d_y - \nabla_y u - b_y^k\right\|_2^2 \tag{19}$$

Thus in equation 19 the first 2 TV terms are solved using a soft thresholding function (shrink operator) while the rest 3 are solved using gradient descent.

Algorithm 2 Split Bregman TV Inpainting

1: Initialize $u^0 = f, and d_x^0 = d_y^0 = b_x^0 = b_y^0 = 0$
2: **while** $\left\|u^k - u^{k-1}\right\|_2 > tol$ **do**
3: $u^{k+1} = GradientDescent(u^k)$
4: $d_x^{k+1} = shrink(\nabla_x u^{k+1} + b_x^k, 1/\lambda$
5: $d_y^{k+1} = shrink(\nabla_y u^{k+1} + b_y^k, 1/\lambda$
6: $b_x^{k+1} = b_x^k + (\nabla_x u^{k+1} - d_x^{k+1})$
7: $b_y^{k+1} = b_y^k + (\nabla_y u^{k+1} - d_y^{k+1})$
8: **end while**

Split Bregman Itteration Results

(a) (b)

(a) (b)

Figure 12: (a) Observation Image, (b) Image with Scratches which has to be Inpainted, (c) Output Image Generated, (d) Best Output Image keeping features.

References

[1] M. Aharon, M. Elad, and A. Bruckstein, "*rmk*-svd: An algorithm for designing overcomplete dictionaries for sparse representation," *IEEE Transactions on signal processing*, vol. 54, no. 11, pp. 4311–4322, 2006.

[2] R. Rubinstein, M. Zibulevsky, and M. Elad, "Efficient implementation of the k-svd algorithm using batch orthogonal matching pursuit," *Cs Technion*, vol. 40, no. 8, pp. 1–15, 2008.

[3] T. Goldstein and S. Osher, "The split bregman method for l1-regularized problems," *SIAM journal on imaging sciences*, vol. 2, no. 2, pp. 323–343, 2009.

[4] T. Ogawa and M. Haseyama, "Image inpainting based on sparse representations with a perceptual metric," *EURASIP Journal on Advances in Signal Processing*, vol. 2013, no. 1, p. 179, 2013.