Image Denoising Pipeline

Sai Himal Allu
Ankit Kataria
Deepesh Pathak

Indian Institute of Technology, Roorkee

14 April 2019

Introduction

- Noise in images mainly occurs during the acquisition, transmission and retrieval of signals
- Denoising refers to the process of eliminating noise from the images
- A non-exhaustive list of standard approaches in the literature are:
  - Filter based methods (Non Local Means)
  - Sparse Coding approaches
  - Introducing an effective prior
  - Low Rank approaches
  - Deep Learning Based Approaches
- The pipeline which we experimented with, uses Sparse Coding and Low rank factorization to produce the denoised image.

Sparse Coding

- Sparsity and density
- Noisy images are usually dense matrices. Denoising can be visualized as an image transform that can "sparsify" the dense matrix.
- Aim is to find a sparse representation of the input data through a linear combination of certain basis elements which are referred to as "atoms".
- These "atoms" need not be orthogonal
- Including redundant "atoms" makes the quality of the sparse representation better.

Low Rank approaches

- Low Rank Matrix Factorization (LRMF) and Nuclear Norm Minimization (NNM)
- We have implemented the former in this pipeline
- Given a matrix Y, the aim is to find a matrix X, which is as close to Y as possible, while being able to be factorized into the product of two low-rank matricces. SVD is a variant of this method
- Low rank factorization can be exploited for the restoration of natural images

Building the optimization equation

- Notation
  - Image Matrix $L \in \mathcal{R}^{n \times N}$
  - Image transform $T \in \mathcal{R}^{m \times n}$
  - Sparse Representation $S \in \mathcal{R}^{m \times N}$
  - Error E
- Final Equation $TL = S + E$
- Block Matching Operator: Takes in a reference patch and compares the top "m" patches that are closest in distance in Euclidean terms. Define a patch to be a column in the Image matrix
  - L = [$l_1, l_2, l_3, l_4 ..... l_N$]
  - BM operator Ki : L $\rightarrow$ $KiL \in \mathcal{R}^{n \times M}$
  - l_{ji} arranged in ascending order of their L2 norm to form the columns of matrix $K_i L$

Building the optimization equation (contd)

- To simplify the calculations, the transform matrix is assumed to be unitary i.e $W^T * W = I_n$; $I \in \mathcal{R}^{N \times N}$
- Final Equation to be optimized

$$\left\{\hat{S}, \left\{\hat{D}_i\right\}\right\} = \underset{S,\{D_i\}}{\operatorname{argmin}} \|TL - S\|_F^2 + \gamma_s^2 \|S\|_0$$

$$+ \gamma_I \sum_{i=1}^{N} \left\{ \|K_i L - D_i\|_F^2 + \theta^2 \operatorname{rank}(D_i) \right\}$$

- Notation
  - $\hat{S}$: Optimal Sparse code matrix of L
  - $\hat{D}_i$: Low-rank approximation of the matched block K_iL
  - The remaining constants are the regularizing parameters

Solving the equation

- Goal: Recover the image L by reconstructing all of its patches l_i from the corrupted input y_i
  - $y_i = l_i + z_i$, where z_i is the additive noise for the i^{th} patch.
- Notation
  - Operator G_i selects the i_{th} column of L such that $G_i L = l_i$
- Input: The corrupted image Y and the initial Transform T
- Initialization: $\hat{U}_0 = Y$ and $\hat{W}_0 = W_0$
- 4 step gradient descent
  - **Sparse Representation**: Given L and assuming a fixed T $\hat{S} = \underset{S}{\text{argmin}} \| TL - S \|_F^2 + \gamma_s^2 \| S \|_0$ solve for S

Solving the equation (contd)

- 4 step gradient descent
  - **Transform Update**: Assuming a fixed S, solve for T
    $\hat{T} = \underset{T}{\text{argmin}} \| TL - S \|_F^2 \quad \text{s.t. } T^T T = I_n$
  - **Low rank approximation**: With the BM operators K_i, we solve for each low-rank approximation D_i
    $\hat{D}_i = \underset{D_i}{\text{argmin}} \| K_i L - D_i \|_F^2 + \theta^2 \, \text{rank}\,(D_i)$
  - **Patch Restoration**: Each of the image patches is now restored with with fixed S, T and D_i. The final equation is
    $\hat{I}_i = \underset{I_i}{\text{argmin}} \| TI_i - \alpha_i \|_2^2 + \gamma_f \| I_i - y_i \|_2^2$
    $$+ \gamma_I \sum_{j \in C_i} \| I_i - D_{j,i} \|_2^2$$
- Aggragation: Each patch is weighted by the reciprocal of the sparsity which is represented by α_i. The patches are combined to obtain the denoised image

Implementation

- Language: Python
- Libraries: Numpy, Scipy,
- Code has been open sourced at Image Denoising Pipeline

Results

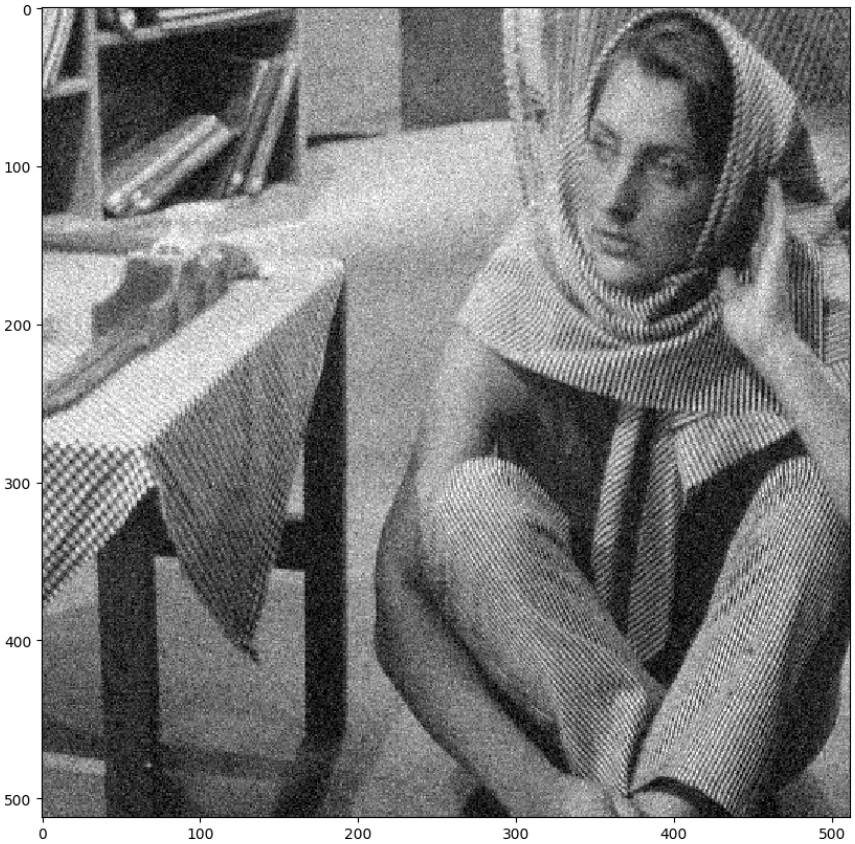
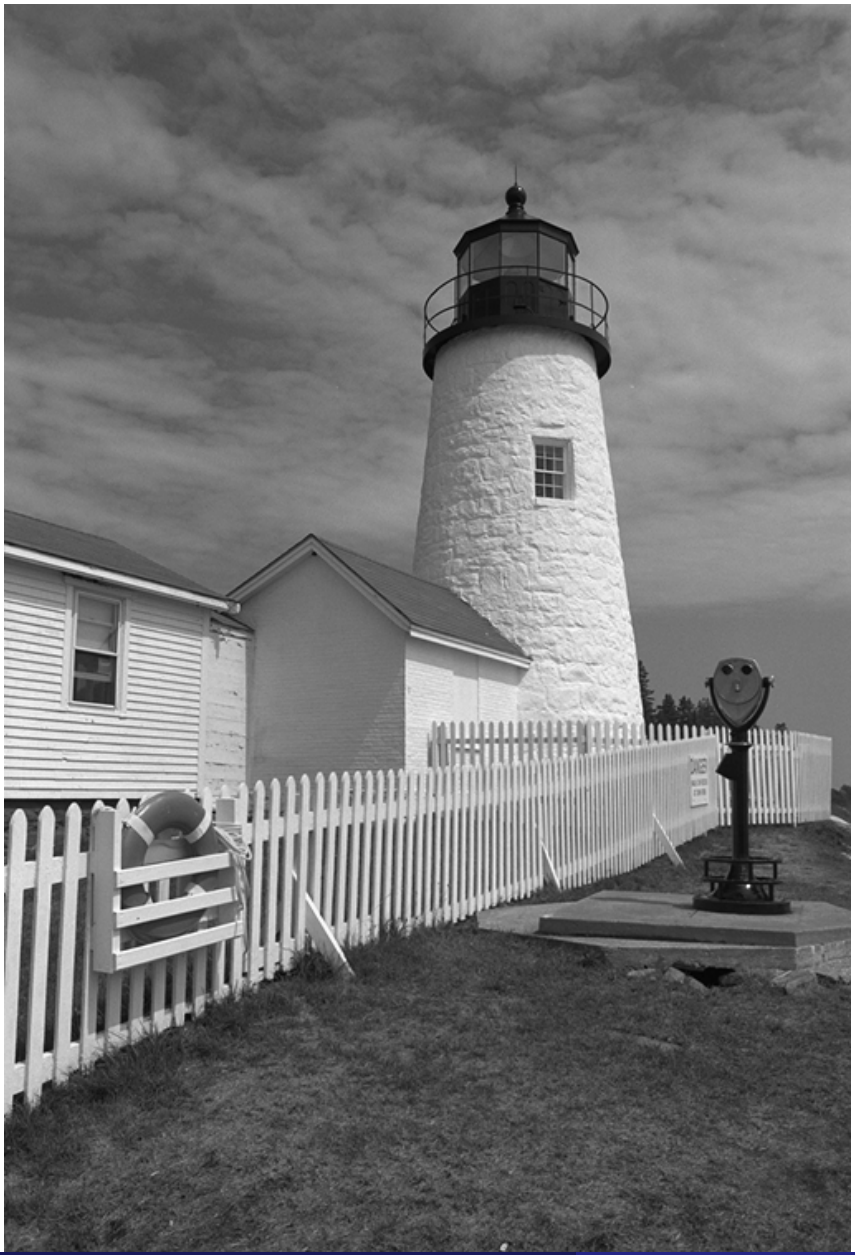


Figure 1: Noisy image

Results (contd)



Evaluation

- Eye test: Although we have been able to eliminate noise in the image, there is some amount of residual blur left in the image. Certain amount of blurring is desirable, since it eliminates the high frequency noise components from the image, but we can work on reducing the blur to a certain extent
- PSNR values: Although we were not able to report results on the Kodak Dataset as was proposed due to the signficant time of operation involved the pipeline, we have tabulated the PSNR values for the results demonstrated here and report good results on them

Image	PSNR
Woman	42
Lighthouse	32

Table 1: PSNR results

Conclusion and Future Work

- The implementation of the whole pipeline takes around 20 minutes for a single image which does not indicate confidence in scaling up this pipeline to a more heavy dataset
- Involvement of expensive mathematical procedures like SVD. Need to focus on finding cheaper alternatives to the above methods.
- Construction of another pipeline to eliminate the additional blur that is introduced as a byproduct of the method implemented.
  Experimented with the above, during the implementation phase of the project, however could not get satisfactory results. Warrants a more exhaustive search over the methods.

Acknowledgement

- The group would like to thank Dr Saumik Bhattacharya for this opportunity and his guidance and support during the entire course of this project.