Monte-Carlo Sure: A Black-Box Optimization for General Denoising Algorithms

Raavi Gupta
Electrical Engineering
Indian Institute of Technology Bombay
200070064

***Abstract*—The original paper [RBU08] uses Stein's unbiased risk estimate (SURE) as an unbiased estimator of mean squared error (MSE). MSE makes use of the original signal to determine the error in the reconstructed signal from the corrupted signal. SURE on the other hand, requires only the knowledge of the noise added, the reconstructed signal and the corrupted signal. On top of the original denoising algorithm proposed by the authors, in this paper SURE has been calculated on some more and the results have been documented. Additionaly, SURE's performance on non-differential denoising algorithms have been discussed and some changes have been done to better approximate the MSE. All the codes and results are made available here**

I. Introduction

II. Background and prior work

We have the noisy data $y \in \mathbb{R}^n$ given by:

$$y = x + b \tag{1}$$

where $x \in \mathbb{R}^n$ represents the vector containing the samples of the unknown deterministic noise-free signal and $b \in \mathbb{R}^n$ denotes the vector containing zero-mean white Gaussian noise of σ^2 variance, respectively.

A denoising operator $f_\lambda : \mathbb{R}^n \to \mathbb{R}^n$ maps the input data y onto the signal estimate:

$$\tilde{x} = f_\lambda(y) \tag{2}$$

SURE corresponding to $f_\lambda(y)$ is a random variable $\eta : \mathbb{R}^n \to \mathbb{R}$ given by:

$$\eta(f_\lambda(y)) = \frac{||y - f_\lambda(y)||^2}{N} - \sigma^2 + \frac{2\sigma^2}{N}\text{div}_y\{f_\lambda(y)\}$$

Lemma II.1. *Let* ***F(y)*** *be an N-dimensional vector function such that for* $\mathcal{E}\{|\delta f_n(y)/\delta y_n|\} < \inf$ *for* $n = 1, ...N$. *Then, under the additive white Gaussian noise assumption, the expressions* $\boldsymbol{F(y)}^T x$ *and* $\boldsymbol{F(y)}^T y - \sigma^2 div\{\boldsymbol{F(y)}\}$ *have the same expectation*

Theorem II.2 (Source: [BL07])**.** *The random variable* $\eta(f_{\lambda(y)})$ *is an unbiased estimator of*

$$MSE(f_\lambda(y)) = \frac{||x - f_\lambda(y)||^2}{N} \tag{3}$$

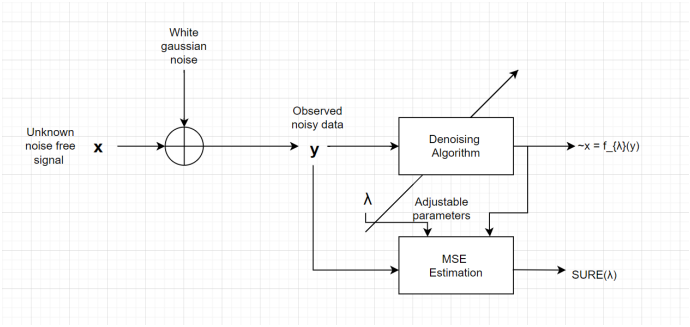

Figure 1. Signal estimate $\tilde{x}$ is obtained by applying the λ-dependent denoising algorithm on the observed data y. The MSE box then computes the estimate SURE (λ) of the MSE between the noise-free x and the denoised as a function of λ, knowing only y and $f_\lambda(y)$.
Image source: [RBU08, Fig. 2]

that is

$$E_b\left(\frac{||x - f_\lambda(y)||^2}{N}\right) = E_b(\eta(f_{\lambda(y)})) \tag{4}$$

Proof. Using the above lemma, we get for the expression of MSE as:
$\mathcal{E}\left(||F(y) - x||^2\right) = \mathcal{E}\left(||F(y)||^2\right) + \mathcal{E}\left(||x||^2\right) - 2\mathcal{E}\left(F(y)^T x\right) = \mathcal{E}\left(||F(y)||^2\right) - 2\mathcal{E}\left(F(y)^T y\right) + 2\sigma^2\mathcal{E}(\text{div}\{F(y)\}) + \mathcal{E}\left(||x||^2\right)$
and $\mathcal{E}\left(||x||^2\right) = \mathcal{E}\left(||y||^2\right) - N\sigma^2$
using which we would arrive at the formulation of SURE mentioned above □

III. Methodology

Algorithm 1: b' is a zero-mean i.i.d. random vector of unit variance

for $\lambda = \lambda_0$, evaluate $f_\lambda(y)$
Build $z = y + \epsilon b'$. Evaluate $f_\lambda(z)$ for $\lambda = \lambda_0$
Compute div $= \frac{b'^T(f_\lambda(z) - f_\lambda(y))}{N\epsilon}$
Using above obtained quantities calculate SURE =0

For calculating the divergence employed in SURE in case of non differential denoising algorithms, the following approaches were employed:

1) Sobel's Operator
   It is a spatial filter that calculates the gradient of an

image at each point, highlighting areas of rapid intensity change.

Kernel used in x direction: $\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$

Kernel used in y direction: $\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$

The kernels are convolved with the image to calculate the gradient in both the directions.

2) Employing central differences instead of forward difference for div calculations
3) Padding the image with a border in case of denoising operators like max/median blurring

IV. Experimental Background

All the analysis has been done on the Barbara image shown below

Figure 2. Original Barbara image

Figure 3. Sample corrupted image with noise std = 20

Figure 4. Denoised image using bilateral filter

Various denoising algorithms were employed including:

1) Redundant Scale-Dependent Wavelet Soft-Thresholding (RSWST): The RSWST filter operates in the wavelet domain, where an image is decomposed into multiple levels of wavelet coefficients. The wavelet coefficients at each level are then thresholded, with the threshold value being determined based on the scale of the coefficients. The RSWST filter uses a scale-dependent thresholding technique that adjusts the threshold value based on the local variance of the coefficients.

$$y = \Psi(\lambda)(x) = \sum_{j=1}^{J} w_j S_{\theta_j}(x) \tag{5}$$

where x is the noisy image, y is the denoised image, $\Psi(\lambda)$ is the RSWST denoising operator with a scale-dependent threshold λ, w_j is a scale-dependent weight that depends on the variance of the wavelet coefficients at scale j, $S_{\theta_j}(x)$ is the soft-thresholding function applied to the wavelet coefficients at scale j, and J is the number of decomposition levels.
The soft-thresholding function:

$$S_{\theta_j}(x) = \text{sign}(x)\max(|x| - \theta_j, 0) \tag{6}$$

where $\text{sign}(x)$ is the sign function, which returns 1 if x is positive, -1 if x is negative, and 0 if x is 0, $|x|$ is the absolute value of x, and θ_j is the scale-dependent threshold that depends on the variance of the wavelet coefficients at scale j and the scale-dependent threshold is given by

$$\theta_j = \lambda\sigma_j\sqrt{\frac{2\log(N)}{N}} \tag{7}$$

where σ_j is the estimated variance of the wavelet coefficients at scale j, and N is the total number of pixels in the image.

Figure 5. Denoised image using RSWST

2) TV Denoising: TV denoising is a type of image denoising method that reduces noise while preserving edges and other image structures. The method is based on the concept of total variation, which measures the amount of variation in an image.
The TV denoising method works by minimizing the total variation of the image subject to a constraint that the denoised image is close to the noisy image. This can be formulated as an optimization problem, which can be solved using various numerical methods, such as gradient descent or proximal algorithms.
The TV denoising problem can be written as follows:

$$\min_{u \in BV(\Omega)} \frac{1}{2}\int_{\Omega}(u(x) - f(x))^2 dx + \lambda \int_{\Omega} |\nabla u(x)| dx \tag{8}$$

where u is the denoised image, f is the noisy image, Ω is the image domain, $BV(\Omega)$ is the space of functions of bounded variation, $\nabla u(x)$ is the gradient of u at position x, and λ is a parameter that controls the strength of the denoising.
The first term in the objective function measures the distance between the denoised image u and the noisy

image f. The second term measures the total variation of the denoised image, which encourages the solution to have smooth regions separated by sharp edges. The parameter λ balances the trade-off between the two terms.

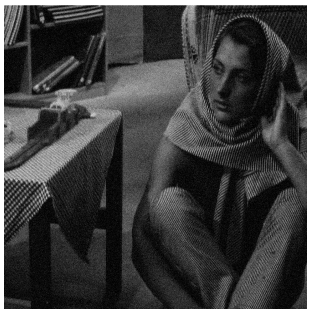

Figure 6. Denoised image using TV denoising

3) Bilateral filter: The bilateral filter computes the filtered image as a weighted average of the neighboring pixels, where the weight depends on both the spatial distance and the intensity difference between the center pixel and its neighbors. The weight function is typically defined as a product of two Gaussian functions: one for the spatial distance and one for the intensity difference. The bilateral filter is defined as:

$$\text{BF}(f)(x) = \frac{1}{W_p} \sum_{y \in \Omega} f(y) w(x, y) \tag{9}$$

where $\text{BF}(f)(x)$ is the filtered image at position x, $f(y)$ is the pixel value of the input image at position y, $w(x, y)$ is the weight between the center pixel x and the neighbor pixel y, and W_p is the normalization constant. The weights are given by:

$$w(x, y) = \omega(||x - y||)\phi(|f(x) - f(y)|) \tag{10}$$

where $\omega(||x - y||)$ is a Gaussian function that depends on the spatial distance between the pixels x and y, and $\phi(|f(x) - f(y)|)$ is a Gaussian function that depends on the difference in intensity values between the pixels x and y.

Figure 7. Denoised image using bilateral filters

4) Median blurring: Median blurring is a non-linear image filtering technique that replaces each pixel in the image with the median of the intensity values in its surrounding neighborhood. It is a type of rank filter that preserves the edges and other structures in the image, while removing the noise and small details.
The median filter works by sliding a window of a fixed size over the image and computing the median value of the pixel values within the window. The center pixel value is then replaced by the computed median value. The median filter can be expressed mathematically as:

$$\text{med}(f(x, y)) = \text{median}\{f(x+i, y+j) : i, j \in \{-k, \ldots, k\}\} \tag{11}$$

where $\text{med}(f(x, y))$ is the median filtered value of the pixel at position (x, y) in the image f, and k is the size of the window or the radius of the filter.

Figure 8. Denoised image using Median blurring

V. Results

Below are the results obtained to track MSE using SURE:

Average error between MSE and SURE		
Denoising Algorithm	Varying std	Varying epsilon
RSWST	0.88333	0.73407
TV Denoising	0.59852	0.43097
Bilateral Filter	0.23059	3.57725
Median Blurring	7856.20	8933.79125

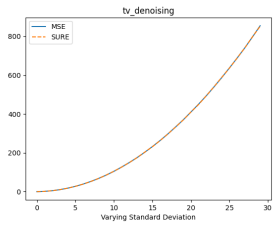

Figure 9. Variation of SURE with std for TV denoising

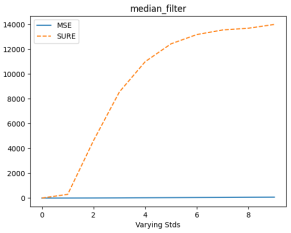

Figure 10. Variation of SURE with std for Median blurring

As can be seen from the above table the error in case of median blurring is easily differentiated from the other three as median blurring is not a differential denoising algorithm while the other three are. To counter this, three different approaches were taken:

1) Sobel's operator:
2) Using forward central differences:

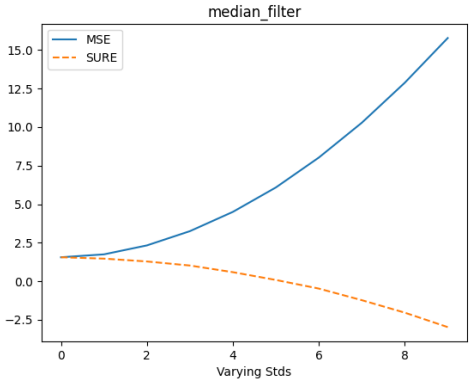

Figure 11. Replication of MSE using Sobel operator to calculate div

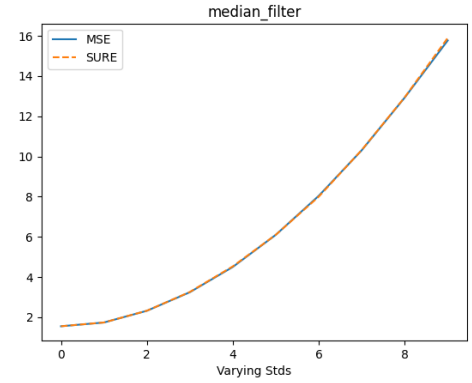

Figure 12. Forward differences

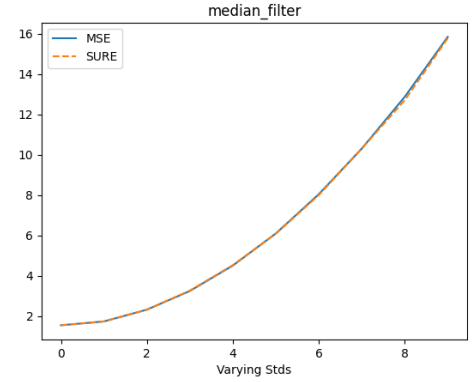

Figure 13. Central difference

As can be seen by employing central diff we are able to do much better as compared to our previous approach to calculate SURE

ACKNOWLEDGEMENT

I would like to thank Prof. Satish Mulleti for teaching this course, without which I would not have been able to complete this project. I would also like to thank the TA Parth involved with this course, who helped me a lot in doing the various assignments, and completing this course.

REFERENCES

[BL07] Thierry Blu and Florian Luisier. "The SURE-LET approach to image denoising". In: *IEEE transactions on image processing : a publication of the IEEE Signal Processing Society* 16 (Dec. 2007), pp. 2778–86. DOI: 10.1109/TIP.2007.906002.

[RBU08] Sathish Ramani, Thierry Blu, and Michael Unser. "Monte-Carlo Sure: A Black-Box Optimization of Regularization Parameters for General Denoising Algorithms". In: *IEEE Transactions on Image Processing* 17.9 (2008), pp. 1540–1554. DOI: 10.1109/TIP.2008.2001404.