Journal of Electronic Imaging

JElectronicImaging.org

Moran's *I* for impulse noise detection and removal in color images

Chih-Cheng Hung
Eun Suk Chang

Chih-Cheng Hung, Eun Suk Chang, "Moran's *I* for impulse noise detection and removal in color images," *J. Electron. Imaging* **26**(2), 023023 (2017), doi: 10.1117/1.JEI.26.2.023023.

Moran's *I* for impulse noise detection and removal in color images

Chih-Cheng Hung[a,b,*] and Eun Suk Chang[a]
[a]Kennesaw State University, Laboratory for Machine Vision and Security Research, Marietta, Georgia, United States
[b]Anyang Normal University, Sino-US Information Processing Joint Research Lab, Anyang, China

Abstract. An approach for impulse noise detection and removal in color images based on Moran's *I* (MI) statistic is proposed. The proposed method consists of detection and removal components and is called Moran's *I* vector median filter (MIVMF). The detection module is able to determine if a pixel is noise or noise-free. If it is a noise pixel, the vector median filter (VMF) will be used to remove the noise. This detection capability meets the so-called "switching" mechanism, which only selects noisy pixels for denoising. Hence, this proposed filter will expedite the processing time with the reduced number of vector calculations in the VMF due to this detection function. This type of detection is achieved with MI index and the indication of one-dimensional Laplacian kernels. We compare the proposed MIVMF with other well-developed vector-type median filters in the literature. Our experimental results show that the proposed filter is not only faster in the filtering process but also efficient in removing random impulse noise with different noise levels in color images. The MIVMF demonstrates a promising denoising result based on the criteria of peak signal-to-noise ratio and structural similarity index metric. With the visualization of processed images, the MIVMF can avoid image blurring, preserve the edge details, and achieve superior noise reduction. [DOI: 10.1117/1.JEI.26.2.023023]

Keywords: impulse noise; Moran's *I* statistic; one-dimensional Laplacian kernels; vector median filter.
Paper 16519 received Jun. 20, 2016; accepted for publication Mar. 30, 2017; published online Apr. 22, 2017.

1 Introduction

The major sources of noise corrupted in most digital images are from the process of image acquisition, quantization, and transmission. As one common example, when an image is transmitted through some wireless mobile networks, it may be ruined with noisy signals from atmospheric disturbances, such as thunder and lightning in the environment.[1] Thus, an image may be degraded such that it will not have the same original quality. The consequence of degraded images can create potential problems for further image processing and analysis. For example, a clustering algorithm for segmenting an image usually measures the relationships in the pixel space by categorizing the pixels into different classes.[2] Therefore, the existence of noise pixels will create different attributes for pixels that originally belonged to the same cluster. This may generate different clustering outcomes. Our goal is to remove noise in color images and restore the image, which, as a result, will be as similar as possible or identical to the original image.

In general, algorithms for removing noise in images can be divided into two methods: one for grayscale images and one for color images. Many state-of-the-art noise removal algorithms have been proposed in the past decades for the grayscale image.[3–7] As digital imaging technology has advanced, the necessity for color image processing has been in great demand. Although grayscale image processing is still needed in specific fields, such as medical imaging on x-ray computed tomography,[8] the wide usage of color and multichannel images is very common in many areas of imaging applications. In this study, we consider color images our primary focus in developing the denoising filter.

Prior to building the algorithm for denoising filters, one of the most important prerequisites is to take on the noise models for color images. Image noise occurs in a wide variety of forms.[1,5] It is very common for noise to contaminate the pixels of images taken by the sensor. During the process of digitization and transmission, noise can be introduced into each pixel wherein one or more bit-errors will be embedded in the pixels of images. The Gaussian noise model and impulse noise model are two widely used models for characterizing the noise information for denoising filters in digital image processing. The Gaussian model is convenient for the simulation of noise to design and test the efficiency of a denoising algorithm. If the noise causes pixel data loss or saturation, the impulse noise model, also called salt-and-pepper noise, will be suitable for the noise characterization. Gaussian noise is used as a model of dark noise and often as a crude model of shot noise, which is governed by the Poisson distribution. The main sources of Gaussian noise in digital images arise during acquisition, for example, sensor noise caused by factors such as poor illumination, high temperature, and transmission noise. There are also different noises caused by devices, such as dark noise, which is due to the thermal fluctuations of stationary charge carriers. Dark noise frequently occurs in image sensors, such as charge-coupled devices. Noise degrades image quality and causes

*Address all correspondence to: Chih-Cheng Hung, E-mail: chung1@kennesaw.edu

difficulty in further processing. Hence, an image denoising algorithm requires an image noise model for improving the signal-to-noise ratio in digital images. In this work, we assume that the impulse noise model is the major type of noise to be detected and removed. Impulse noise randomly and sparsely corrupts pixels to two intensity levels, relative high or relative low, compared with its neighboring pixels.

There exist several different formulations for the impulse noise model, and each formulation has its own characteristic in generating how the density level of noise will be assigned and how three color channels will be corrupted with a certain level of probability.[9–11] Let us assume that the density level of noise probability is δ and k value is the color channel index for red, blue, and green. For convenience, we assume that the red, green, and blue channels are assigned 1, 2, and 3, respectively, with an 8-bit pixel image for the following discussion. If δ is assigned 0.2, then 20% of the entire pixels of a given image is noise. The first impulse noise model, which is uncorrelated impulsive noise,[10] is formulated as follows:

$$x^k = \begin{cases} r^k_{\text{SAP}} \text{ with probability } \delta \\ \text{or} \\ o^k \text{ with probability } 1-\delta \end{cases} \quad \text{where } k = 1, 2, \text{ or } 3, \tag{1}$$

where r^k_{SAP} stands for the random vector that represents the impulse noise corruption with $r^k_{\text{SAP}} \in [0]$ or $r^k_{\text{SAP}} \in [255]$ (with equal probability) and o^k represents the original color vectors. The subscript SAP represents the salt-and-pepper noise. Those two vectors result in the final outcomes x^k of the corrupted image with impulse noise restricted to 0 or 255. This model is called a "salt-and-pepper" impulse noise type. This model is limited to only one channel corruption at a time for all three channels. Hence, we use lower case symbols r^k_{SAP}, o^k, and x^k to represent this model.

The second impulse noise model is based on the correlation of impulse noise distribution among the color channels. This model can be expressed as follows:

$$X = \begin{cases} [o^1, o^2, o^3] \text{ with probability } 1-\delta \\ [r^1_{\text{SAP}}, o^2, o^3] \text{ with probability } \delta_1 \text{ for channel } 1 \text{ (under } \delta) \\ [o^1, r^2_{\text{SAP}}, o^3] \text{ with probability } \delta_2 \text{ for channel } 2 \text{ (under } \delta) \\ [o^1, o^2, r^3_{\text{SAP}}] \text{ with probability } \delta_3 \text{ for channel } 3 \text{ (under } \delta) \\ [r^1_{\text{SAP}}, r^2_{\text{SAP}}, r^3_{\text{SAP}}] \text{ with probability } \delta_4 \text{ for all channels (under } \delta) \end{cases}, \tag{2}$$

where X represents a pixel vector and r^i_{SAP} and o^i are similarly defined as in Eq. (1). δ_1, δ_2, and δ_3 are the respective probabilities of each channel corruption (these parameters were set up as $\delta_1 = \delta_2 = \delta_3 = \delta_4 = 0.25$ in Ref. 10). For example, if we assume that the noise level is 0.1 (i.e., 10% noise density), the δ value is 0.1. Each individual channel δ_i will be given 0.25 probabilities under the overall 10% probability. Consequently, the combination of single channel corruptions from δ_1, δ_2, and δ_3 and all channel corruption will add up to 0.1 probability in total. The problem with this model is that it limits the noise corruption either on a single channel only or three channels simultaneously but does not permit two channels for the noise corruption simultaneously. This may not be realistic in practical applications.

The noise model that we used in our research is not limited to one channel for noise corruption. The noise signal used is also not limited to two relatively high and low values (i.e., values close to 0 or 255). Our model assumes random-valued noise corruption on the multiple channels simultaneously with a randomized noise level from 0 to 255. We assume that all channels are corrupted with the probability δ. The impulse noise model we used is shown in Eq. (3), where the subscript rv in r^i_{rv} stands for the random value. This model allows any combination of channels (can be 1, 2, or 3) to be corrupted with impulse noises. This is a more general impulse noise model compared with noise models 1 and 2. As a result, the comparison in terms of visual perception of noise corruption level is more severe than noise model 2. Please note that noise model 1 as shown in Fig. 1(b) looks more noisy than our model in Fig. 1(d) due to the difference in the three-channel noise (i.e., color) and salt-and-pepper. The latter makes dark and white noises more prominent compared with the color.

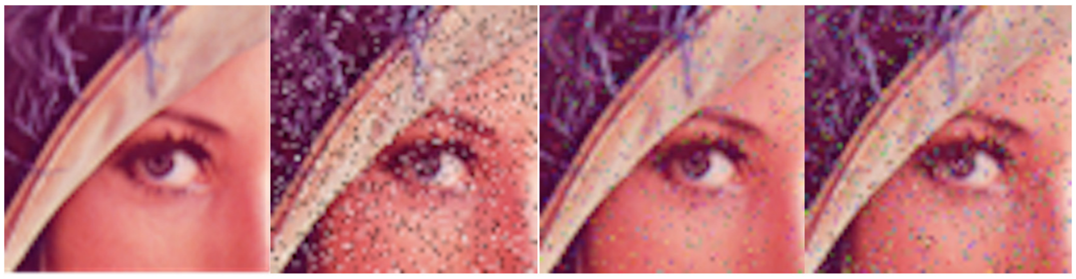


Fig. 1 A comparison of different impulse noise models: (a) an original image, (b) corrupted with the noise model 1, (c) corrupted with the noise model 2, and (d) corrupted with the noise model 3. The probability δ is set to 10% for all noise models. All testing simulations in this paper will be based on noise model 3.

$$\mathbf{X} = \begin{cases} [o^1, o^2, o^3] \text{ with probability } 1-\delta \\ \text{or} \\ [r^1_{\text{rv}}, r^2_{\text{rv}}, r^3_{\text{rv}}] \text{ with probability } \delta \end{cases}. \tag{3}$$

There are many state-of-the-art developments and publications in the literature for impulse noise removal in color images. Similar to grayscale methods, many of them have their own advantages and limitations in terms of performance issues, such as edge-preserving capability and efficiency on noise removal. One of the important functions that has been used in the grayscale methods for the median filter (MF) is called "switching based"-MFs.[4–6] This type of MFs selectively suppresses noise pixels, leaving those uncorrupted pixels intact. It can preserve the edge details and avoid blurring the image. This mechanism is also used in some versions of the vector median filter (VMF) for color images.[10,12,13] Our proposed idea on Moran's *I* (MI) (statistic) and four one-dimensional (1-D) Laplacian kernels is designed to improve the performance on the capability of noise detection, which will then be used for noise removal function. Our proposed filter is also a type of filter with a switching mechanism. MI is one of the oldest indicators of spatial autocorrelation frequently used in social sciences, such as geography and sociology. Based on our research in the literature, two groups of researchers have proposed the MI statistic in their work. Chen et al.[14] used the MI statistic for measuring the image quality of reconstructed images. Chuang and Huang[15] assume that the byte of a pixel in an image can be divided into two parts, signal and noise; the noise occupies the lower bits of a byte. The MI and join-count statistics then examine the noise bits to filter them out. In our study, the MI statistic is used as a measure in a defined window for detecting noise pixels. The motivation for this study is to measure the strength of spatial autocorrelation of pixels in the local neighborhood of the image. MI indicates the different spatial structures of the smooth and rough surfaces, which provide an index for impulse noise determination and preserve original noise-free pixels. In Sec. 3, we will introduce MI and its application on impulse noise detection in detail.

The rest of this paper is organized as follows. The related work in Sec. 2 will briefly review research in impulse noise removal of color images and present our initial thought and motivation for improving contemporary filtering methods for color images. A comparison of the classical MF and VMF is given in this section based on our experimental results. In Sec. 3, the proposed algorithm with spatial autocorrelation associated with MI and Laplacian kernels is provided. Section 4 describes the simulation results, and Sec. 5 gives the concluding remarks and future work.

2 Related Work

2.1 *Classical Median Filter and Vector Median Filter*

The classical MF is an efficient filtering algorithm for removing impulse noise in gray-level, color, and multichannel images. It is one of the most common nonlinear type filters. Due to its distinctive property in the impulse response of the MF, this filter has been widely used in suppressing impulsive noise.[16] Many authors have analyzed the statistical properties of the filter, and some modifications and generalizations of the MF have been introduced.[16] This MF is a simple and efficient algorithm that replaces the center pixel with the median value of the corresponding neighborhood window, such as a size of 3×3. A general MF is formulated as

$$\check{f}(x,y) = \text{med}\{f(x+s, y+t) | (s,t) \in W) \\ \{-\lfloor m/2 \rfloor \le s \le +\lfloor m/2 \rfloor, -\lfloor m/2 \rfloor \le t \le +\lfloor m/2 \rfloor\}, \tag{4}$$

where $\check{f}$ is the filtered image, (x, y) is the location of a pixel, W is the neighborhood window in the image, and the s and t values are determined by the window size m and the floor function (i.e., $\lfloor m/2 \rfloor$) for integer truncation.

Even though this filtering algorithm was originally developed for the grayscale image, it can intuitively be used in color images with a component-wise extension. The so-called marginal standard median filter (MSMF)[8,17,18] deals with each red, green, and blue channel separately, by calculating the median value of each channel within a neighborhood window. The restored image is then constructed by combining all RGB channels together, as shown in Fig. 2. Consequently, the filtered pixel will have the median value for each channel as a component in the pixel vector. This filter considers the median value in each separated channel without considering the correlation that may exist between color channels. As a result, the restored image may have color distortion (artifacts).[8,18]

Astola et al.[16] proposed a VMF for impulse noise removal for color images in 1990. The VMF is based on ordering the vectors within the neighborhood window by calculating the cumulative pairwise distance and taking the vector corresponding to the lowest-ranked distance as the vector median value for the output. The VMF is formulated as in Eq. (5)

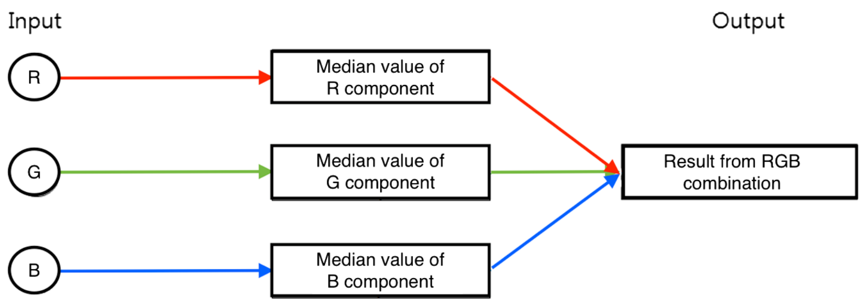

Input
Output
R
G
B
Median value of R component
Median value of G component
Median value of B component
Result from RGB combination

Fig. 2 The MSMF applies to each component separately. A pixel consists of RGB components. The median value of each component is calculated, and the output is a new vector with the median value in each of the RGB components.

where the norm L can be implemented as either L_2 (i.e., Euclidean distance) or other different norms

$$X_{\mathrm{VMF}} = \arg \min_{X_j \in W} \sum_{k=1}^{m} \|X_j - X_k\|_L, \tag{5}$$

where X_j (X_k) represents a pixel vector in the window and X_{VMF} is the pixel vector with the lowest-ranked distance. So far, we reviewed two basic filtering algorithms used for impulse noise removal. Most of the state-of-the-art filters evolved from these two basic filters. Our filter is also motivated by these two basic filters. In Sec. 2.2, we give a brief comparison of these two basic filters from the perspective of both the performance and time complexity of the filters. Please note that we will denote the MSMF as MF since it is just an extension of the traditional MF for color images.

2.2 *Brief Comparison Between the Median Filter and Vector Median Filter*

Prior to moving into constructing new algorithms for impulse noise filtering, we briefly compared the MF and VMF for their similarities and differences on the filtering performance. Many publications have pointed out the advantage of the VMF for multichannel data due to the correlation that exists between the multichannel data. However, in some color images, such an advantage may not present. For example, if we take a 3×3 neighborhood window in a color image with all eight pixel values [255, 255, 0] and one noise pixel [0, 255, 255], both MF and VMF will have identical denoising results. This might be the reason that the MF shows a very competitive result compared with those of VMF in the following experiments based on the peak signal-to-noise ratio (PSNR) and structural similarity index metric (SSIM) measures.

Multiple sample images were tested to compare the denoising quality and time complexity. To assess the denoising quality, we used the PSNR[19] and SSIM[20] as the quantitative measure. The PSNR is calculated below

$$\mathrm{PSNR} = 10 \times \log\left(\frac{255^2}{\mathrm{MSE}}\right), \tag{6}$$

where

$$\mathrm{MSE} = \sum_{i=1}^{M} \sum_{j=1}^{N} \frac{(I_{ij} - \check{I}_{ij})^2}{M \times N}, \tag{7}$$

and M and N are the width and height of an image, I_{ij} for an individual pixel of the noise-free image, and $\check{I}_{ij}$ is the corresponding pixel of the filtered image. The higher value indicates a better denoising result. The SSIM is designed to measure image quality considering human subjectivity in the context of luminance, contrast, and structural similarity.[20] The comparison between original image x and filtered image y can be calculated using the following equation:

$$\mathrm{SSIM}(x, y) = \frac{(2\mu_x 2\mu_y + C_1)(2\mu_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)}, \tag{8}$$

where μ_x and μ_y are the average intensities and σ_x, σ_y are the standard deviations corresponding to images x and y, respectively, and μ_{xy} is the covariance. The values C_1 and C_2 are constants determined by the dynamic range value.

To conduct a comparison study for MF and VMF, Lena, pepper, and baboon images (Fig. 3) were tested with 5%, 10%, and 15% noise levels. The size of the filtering neighborhood window is 3×3 and each image is 512×512 pixels. In terms of image complexity, we calculated the entropy for each image using the below equation, which is based on the average (f) of three color channels

$$H(f) = -\sum_{i=0}^{L-1} p_i \log_2 p_i, \tag{9}$$

where $H(f)$ is the sum of entropy for each gray-level probability, p_i, in the histogram of the average of three color channels, f, and $L - 1$ is the maximum gray level for an image. (Here, L is the total number of gray levels.) In general, as an image complexity increases, its entropy value also increases.

Our empirical study on the MF and VMF revealed some interesting outcomes. As shown in Tables 1–3, the overall results from VMF are not significantly different from those of MF. This is shown in Table 3 for the baboon image, in particular, which has the highest entropy among the three images tested. In terms of time complexity, VMF is more than 10 times slower than MF in all test cases. However, VMF is generally considered to be more appropriate for color image processing because it considers inherent correlation between the red, green, and blue color channels of the image and prevents image color distortion.[8,10,17,18] Even with some advantages over MF, the essential problem of slow filtering speed is the major

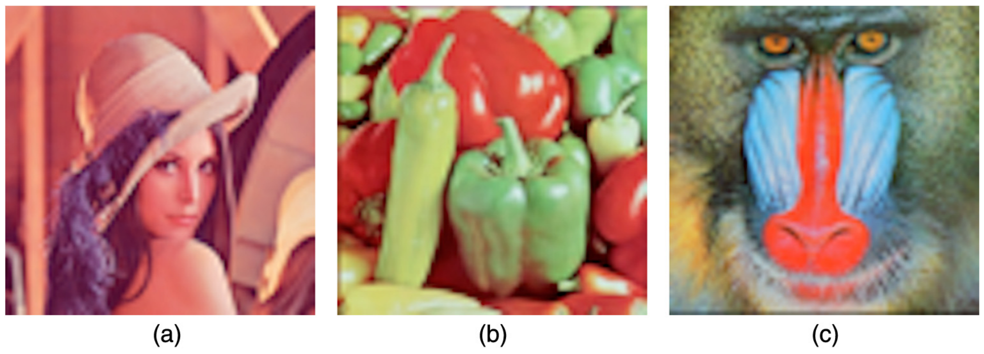


Fig. 3 Sample images tested: (a) Lena, (b) pepper, and (c) baboon. The image size is 512×512.

Table 1 A comparison of MF and VMF for Lena image with entropy value 15.12.

Time spent	MF (s)	VMF (s)
5% noise	**0.281**	3.67
10% noise	**0.288**	3.73
15% noise	**0.284**	3.74
PSNR	MF	VMF
5% noise	**28.6**	28.3
10% noise	**27.9**	27.8
15% noise	27.0	**27.2**
SSIM	MF	VMF
5% noise	**0.984**	0.983
10% noise	**0.981**	0.980
15% noise	0.976	**0.977**

Note: s denotes seconds.

Table 2 A comparison of MF and VMF for peppers image with entropy value 15.30.

Time spent	MF (s)	VMF (s)
5% noise	**0.27**	0.356
10% noise	**0.29**	0.357
15% noise	**0.28**	0.360
PSNR	MF	VMF
5% noise	31.4	**31.4**
10% noise	30.2	**30.4**
15% noise	28.7	**29.1**
SSIM	MF	VMF
5% noise	**0.990**	0.982
10% noise	0.987	**0.987**
15% noise	0.982	**0.984**

Note: s denotes seconds.

weakness of VMF although some high-speed versions have been proposed.[21] If we consider the VMF for the real-time application, its time complexity needs be improved. From this perspective, our proposed filter design will not only possess the advantages of the VMF but also improve its filtering process speed.

Table 3 A comparison of MF and VMF for baboon image with entropy value 15.90.

Time spent	MF (s)	VMF (s)
5% noise	**0.283**	3.77
10% noise	**0.276**	3.78
15% noise	**0.299**	3.75
PSNR	MF	VMF
5% noise	**18.3**	18.0
10% noise	**18.0**	17.8
15% noise	**17.7**	17.5
SSIM	MF	VMF
5% noise	**0.919**	0.915
10% noise	**0.911**	0.906
15% noise	**0.902**	0.897

Note: s denotes seconds.

2.3 *Brief Review on Various Vector Median Filters*

Since VMF was introduced in the early 1990s, many variations of VMF have been proposed, and some of them use different distance measures.[8,9,18,22] The basic vector directional filter (BVDF) utilizes the angle between two pixel vectors within the window.[8] The summation of angular distances used in this filter is calculated below

$$\alpha_i = \arg\ \min \sum_{j=1}^{9} A(X_i, X_j) \qquad i = 1, 2, 3, \dots, 9, \tag{10}$$

where $A(X_i, X_j)$ denotes the angle between pixel vectors X_i and X_j using the arc cosine function from the following equation:

$$A(X_i, X_j) = \cos^{-1}\left(\frac{X_i X_j^T}{|X_i||X_j|}\right). \tag{11}$$

The problem with the basic vector median approach is that it can alter some pixels even if there are possibilities that those pixels are actually noise free. Consequently, the excessive filtering causes blurring effects on the denoised image. To solve this problem, the center weighted vector median filter (CWVMF)[8,9] was proposed to use a weight value for the center pixel in the window. This algorithm, however, lacks an impulse noise detection scheme. Consequently, the adaptive center weighted vector median filter (ACWVMF)[18,23] was proposed. The main difference between the ACWVMF and other filters introduced so far is that it has noise detection capability, instead of performing noise removal on every pixel. This ACWVMF filter provides a switching mechanism that differentiates between impulse noise and noise-free pixels. In the ACWVMF, the noise detection step, which serves as a switching mechanism, is based on the concept of aggregated distances assigned to the pixels in the filtering window.[18] The difference between the accumulated distances assigned to the central pixel and to the pixel with the lowest rank serves as an indicator of the noise. If the indicator is greater than (or equal) a fixed threshold, the output of the ACWVMF is a weighted mean of the central pixel of the filtering window and the vector median of its samples. Otherwise, the pixel is noise-free. In our proposed algorithm, MI statistic has a better indication than the mechanism used in the ACWVMF for noise or noise-free pixel. This statement will be verified later in the experimental results, as shown in Table 21.

Another approach is based on the "peer group" concept.[21,24,25] This type of filter excludes corrupted pixels in the window to calculate the median vector value. The peer group, in short, is the group of pixels in the filtering window that minimizes the total sum of distances from the center pixel of the group to adjacent pixels. Therefore, the peer group vector median filter (PGVMF) is based on the trimmed sum of distances. If the number of pixels in the peer group (denoted by α) is equal to 9 in the window size of 3×3, it will be identical to the basic VMF algorithm (Fig. 4). The replaced pixel vector is determined by the pixels located within a peer group of pixels, which shows the minimum distance or minimum dispersion.

Compared with various filters introduced so far, the robust switching vector median filter (RSVMF)[12] has a solid noise detection algorithm that enables a filtering process; if a pixel is determined as a noisy pixel, the RSVMF will use the VMF to remove the noise pixel. Otherwise, the pixel value remains unchanged. The RSVMF works in a similar way as VMF except for the following modification:

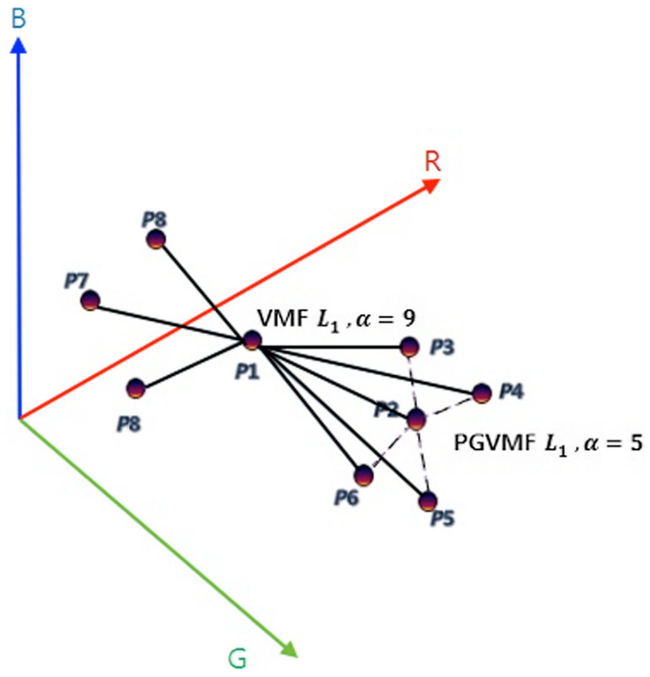


Fig. 4 A comparison of VMF and PGVMF.[25] In this example, *P*1 is the vector median because *P*1 has the minimum distance from itself to all others with the VMF. For PGVMF, peer group vector median is *P*2 because alpha (α) value is 5; *P*2 is the minimum distance.

Table 4 Different filters being compared in this study.

Filter type	Abbreviation
Vector median filter	VMF
Basic vector directional filter	BVDF
Center weighted vector median filter	CWVMF
Adaptive center weighted vector median filter	ACWVMF
Peer group vector median filter	PGVMF
Robust switching vector median filter	RSVMF

$$\begin{cases} \text{if } d_{\text{center}} \le \alpha \cdot \text{med}(d_1, d_2, d_3, \ldots, d_m) \text{ then no filtering} \\ \quad \text{otherwise, the filtering is same as in VMF} \end{cases}, \tag{12}$$

where m is the total number of pixels in a window, d_{center} is the cumulative Euclidean distances from the center pixel to pixel $1,2,\ldots,m$ in the window, and α is a constant value. The med function is to select median value from the cumulative distances of $d_1, d_2, d_3, \ldots, d_m$ using the way we calculate the cumulative distance in the VMF.

Another denoising method is to process vector-based filtering through the trimming scheme.[26] The basic idea of trimming schemes is to select a group of pixels in the filtering window that may exclude noise pixels through the ranking order. In a sense, this filter is very similar to the PGVMF. However, the RSVMF has some limitation on removing the noise pixels if the noise level is increased to a certain percentage, such as 30% noise that is used in our experiments.

All of the algorithms reviewed so far have their own unique characteristics and limitations for filtering impulse noise through vector-based processing. There are several different variations of algorithms reviewed above to improve the result in such a way that the filter can preserve edge details and minimize blurring effects.[13] A comprehensive survey on impulse noise removal filters was given in Ref. 10. Our objective is to design an efficient algorithm for detection and removal of the random-valued impulse noise in color images based on the fundamentals of MI statistics and Laplacian kernels. Our preliminary study on multispectral images shows that this filter is effective in removing impulse noise.[27] In summary, the vector-type MFs listed in Table 4 were tested and compared in this study.

3 Proposed Algorithm: Moran's I Vector Median Filter

Our proposed algorithm for noise detection and removal in color images consists of three major components: MI for spatial autocorrelation, four 1-D Laplacian kernels, and the VMF. The first two components are used for noise detection in the algorithm, and the VMF is used for denoising.

3.1 *Moran's I for Spatial Autocorrelation*

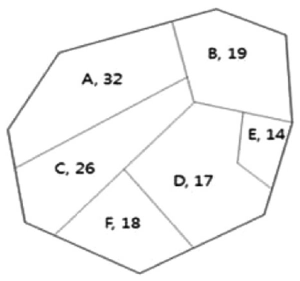
The concept of spatial autocorrelation can be traced back to the first law of geography—"Everything is related to everything else, but near things are more related than distant things." By definition, spatial autocorrelation is clarified as follows: "Spatial autocorrelation refers to the fact that the value of a variable at one point in space is related to the value of that same variable in a nearby location."[28] In fact, the spatial information concept has long been used in pattern recognition and image analysis with different types of formats. To determine the spatial autocorrelation, Moran introduced MI to check the degree of spatial autocorrelation in areal data.[29] MI is calculated by the following equation, and its range is between -1 and $+1$:

$$\text{Moran's}\, I = \frac{m\sum_i^m \sum_j^m \omega_{ij}(y_i - \bar{y})(y_j - \bar{y})}{\sum_i^m \sum_j^m (\omega_{ij}) \sum_i^m (y_i - \bar{y})^2}. \tag{13}$$

Here, we assume that there are m regions, and ω_{ij} is a measure of the spatial proximity between regions i (with value y_i) and j (with value y_j). The parameter $\bar{y}$ is the mean value of m regions. The interpretation of the I value is categorized into two specific spatial patterns. If the value closes to $+1$, it indicates a strong homogeneous spatial autocorrelation where local similarity or homogenous spatial patterns exist. A value nearing -1 indicates the strong heterogeneous spatial autocorrelation, but it is a negative pattern in which heterogeneous values are grouped together (e.g., a mixture of very high and very low values). This negative spatial autocorrelation is unusual in nature. For example, high-income households are not usually located in urban twilight zones. If the I value is zero, there is an absence of any strong spatial autocorrelation or no distinct spatial pattern, such as checkerboard.[28] The calculation of MI can be exemplified with the areal data in Fig. 5(a), which is similar to the example in Ref. 27.

To find out the weight value of ω_{ij}, it is necessary to check the connectivity between regions i and j (i.e., immediate neighbors). For example, region A has connectivity with B and C. Region D has connectivity with B, C, E, and F, as shown in Fig. 5(a). Hence, a binary connectivity weight matrix [shown in Fig. 5(b)] can be constructed corresponding to the data in Fig. 5(a).

From the denominator of Eq. (13), $\sum_i^m (y_i - \bar{y})^2$ represents the variance of the regional value. As a result, m is replaced by 6 [since there are six regions in Fig. 5(a)], MI is calculated as (the detail is omitted here)



(a)

$$\omega = \begin{array}{c|cccccc} & A & B & C & D & E & F \\ \hline A & 0 & 1 & 1 & 0 & 0 & 0 \\ B & 1 & 0 & 1 & 1 & 1 & 0 \\ C & 1 & 1 & 0 & 1 & 0 & 1 \\ D & 0 & 1 & 1 & 0 & 1 & 1 \\ E & 0 & 1 & 0 & 1 & 0 & 0 \\ F & 0 & 0 & 1 & 1 & 0 & 0 \end{array}$$

(b)

Fig. 5 (a) An area of six regions that are represented by A, B, C, D, E, and F with a value associated with each region[26] and (b) corresponding binary connectivity matrix.

$$6\sum_i^6 \sum_j^6 \omega_{ij}(y_i - \bar{y})(y_j - \bar{y}) = 6 \times 100,$$

$$\sum_i^6 \sum_j^6 (\omega_{ij}) \sum_i^6 (y_i - \bar{y})^2 = 18 \times 244.$$

Hence

$$\text{Moran's}\, I = \frac{6(100)}{18(244)} = 0.1488. \tag{14}$$

From the above result, the I value gives a small positive spatial autocorrelation for the given data.

The implementation of MI for noise detection in color images can be done with the same procedure as shown in the example above. The difference now is that a region used in the previous example will be replaced by a pixel in the filtering window for impulse noise detection. An example is given in Table 5, which shows a 3×3 filtering

Table 5 (a) A hypothetical example of 3×3 filtering window (one channel only), (b) its pixel numbering scheme for the pixels in (a), and (c) the binary connectivity (i.e., adjacency) matrix for (a) that shows which pixel is adjacent (marked by 1) to pixel number from 0 to 8. Otherwise, 0 will be used for nonadjacency. For example, for pixel 0, pixels 1, 3, and 4 are adjacent to it.

(a)

$$\left\{\begin{matrix} 25 & 25 & 25 \\ 25 & 200 & 25 \\ 25 & 25 & 25 \end{matrix}\right\} \quad \text{for one channel}$$

(b)

Pixel 0	Pixel 1	Pixel 2
Pixel 3	Pixel 4	Pixel 5
Pixel 6	Pixel 7	Pixel 8

(c)

Pixel number	0	1	2	3	4	5	6	7	8
0	0	1	0	1	1	0	0	0	0
1	1	0	1	1	1	1	0	0	0
2	0	1	0	0	1	1	0	0	0
3	1	1	0	0	1	0	1	1	0
4	1	1	1	1	0	1	1	1	1
5	0	1	1	0	1	0	0	1	1
6	0	0	0	1	1	0	0	1	0
7	0	0	0	1	1	1	1	0	1
8	0	0	0	0	1	1	0	1	0

window where the intensity range of a given pixel is between 0 and 255 for an 8-bit color image. Based on the 8-connectivity concept,[1] we can construct a weight matrix similar to Fig. 5(b). The weight matrix is given in Table 5. Please note that only one channel is shown here.

Based on the weight matrix formulated, for each color channel of RGB, MI can be calculated to check the degree of spatial autocorrelation. If all MI values from three channels have near $+1$ value, then all the pixels in the window are relatively similar. In other words, if any MI value from three channels is close to -1, there is a high probability that impulse noises exist in the filtering window. The threshold value ε_0 for determining the noise area is set up through our experiments, which will be explained in detail in Sec. 3.3.

255	0	0
0	0	0
0	0	255

(a)

255	0	0
0	255	0
0	0	255

(b)

0	0	0
–1	2	–1
0	0	0

(c)

0	–1	0
0	2	0
0	–1	0

(d)

0	0	–1
0	2	0
–1	0	0

(e)

–1	0	0
0	2	0
0	0	–1

(f)

Fig. 6 A hypothetical example shows a line with noise in (a) with 0 as the noise in the center pixel and without noise in (b). Four 1-D Laplacian kernels with a size of 3×3: (c) horizontal, (d) vertical, (e) 45-deg, and (f) 135-deg directions.

3.2 *Four One-Dimensional Laplacian Kernels*

By means of a single detector based on MI index, it may not be robust to detecting impulse noise precisely. This is because MI measures a region's spatial autocorrelation. In other words, MI focuses on all pixels of the filtering window, not the central pixel itself in the window. Hence, it is required to have the next level detector for impulse noise. The application of Laplacian kernels is one of the precise and simple detecting algorithms for impulse noise. In fact, several median-based impulse noise detectors often misclassify a noise-free pixel into the noise category.[4,17] In our proposed filter, if only MI is used for the detection, there is a chance that pixels located on the lines and edges can be categorized as impulse noise also. Using the Laplacian kernels as the second-level detection will eliminate this problem.

Table 6 Threshold testing with Lena image: entropy 15.12.

PSNR	$\varepsilon_0 = -0.2$	$\varepsilon_0 = -0.1$	$\varepsilon_0 = 0.0$	$\varepsilon_0 = 0.1$	$\varepsilon_0 = 0.2$	$\varepsilon_0 = 0.3$
5% noise	27.49	**28.95**	28.77	28.58	28.40	28.34
10% noise	23.87	27.51	**27.92**	27.90	27.84	27.81
15% noise	20.95	25.88	27.11	**27.22**	27.20	27.19
SSIM	$\varepsilon_0 = -0.2$	$\varepsilon_0 = -0.1$	$\varepsilon_0 = 0.0$	$\varepsilon_0 = 0.1$	$\varepsilon_0 = 0.2$	$\varepsilon_0 = 0.3$
5% noise	0.9829	**0.9854**	0.9839	0.9830	0.9825	0.9824
10% noise	0.9569	0.9569	**0.9806**	0.9802	0.9799	0.9799
15% noise	0.9138	0.9722	**0.9774**	0.9771	0.9769	0.9769

Table 7 Threshold testing with peppers image: entropy 15.30.

PSNR	$\varepsilon_0 = -0.2$	$\varepsilon_0 = -0.1$	$\varepsilon_0 = 0.0$	$\varepsilon_0 = 0.1$	$\varepsilon_0 = 0.2$	$\varepsilon_0 = 0.3$
5% noise	28.16	31.54	**31.93**	31.77	31.55	31.44
10% noise	23.38	28.80	30.41	**30.50**	30.44	30.39
15% noise	20.48	26.19	28.80	**29.11**	29.10	29.10
SSIM	$\varepsilon_0 = -0.2$	$\varepsilon_0 = -0.1$	$\varepsilon_0 = 0.0$	$\varepsilon_0 = 0.1$	$\varepsilon_0 = 0.2$	$\varepsilon_0 = 0.3$
5% noise	0.9839	**0.9921**	0.9913	0.9903	0.9895	0.9891
10% noise	0.9520	0.9855	**0.9883**	0.9879	0.9874	0.9871
15% noise	0.9036	0.9727	0.9838	**0.9840**	0.9838	0.9837

In a hypothetical example as shown in Figs. 6(a) and 6(b), we assume this image patch is a line with a direction in 135 deg. We also assume that the patch is identical in three channels. MI value is 0.079 in (a), and the center pixel is noise. Based on our proposed algorithm, if we set the threshold value ε_0 to 0.0, the center pixel will be detected as a noise pixel (MI value is not less than the threshold) and will go through the denoising process. However, the minimum response from four 1-D Laplacian kernels is 0. Based on the algorithm in Ref. 4, the response is not considered a noise pixel since its value is so small (i.e., 0). The reason is that we usually set the threshold value T_0 greater than 0 for the response of Laplacian kernels. Hence, MI can overcome the weakness of the 1-D Laplacian kernels in this example. On the other hand, if a line is given as shown in (b), MI value is −0.087. Based on our proposed algorithm, if we set threshold value ε_0 to 0.0, with the second level of detection (i.e., four 1-D Laplacian kernels), it will be detected as a noise-free pixel based on step 4 in our proposed algorithm (the condition ε_0 is satisfied, but the condition T_0 is not satisfied). The minimum response from four 1-D Laplacian kernels is 0 (with the 135-deg direction of the 1-D Laplacian kernel); based on the algorithm in Ref. 4, the response is not considered a noise pixel since its value is small (i.e., 0). This center pixel will be considered a

Table 8 Threshold testing with baboon image: entropy 15.90.

PSNR	$\varepsilon_0 = -0.2$	$\varepsilon_0 = -0.1$	$\varepsilon_0 = 0.0$	$\varepsilon_0 = 0.1$	$\varepsilon_0 = 0.2$	$\varepsilon_0 = 0.3$
5% noise	**19.55**	18.93	18.39	18.11	17.99	17.97
10% noise	17.98	17.98	**18.00**	17.84	17.76	17.75
15% noise	16.66	**17.67**	17.65	17.56	17.52	17.51
SSIM	$\varepsilon_0 = -0.2$	$\varepsilon_0 = -0.1$	$\varepsilon_0 = 0.0$	$\varepsilon_0 = 0.1$	$\varepsilon_0 = 0.2$	$\varepsilon_0 = 0.3$
5% noise	**0.9431**	0.9303	0.9205	0.9161	0.9146	0.9144
10% noise	0.9175	**0.9177**	0.9103	0.9069	0.9057	0.9055
15% noise	0.8894	**0.9041**	0.9000	0.8978	0.8972	0.8971

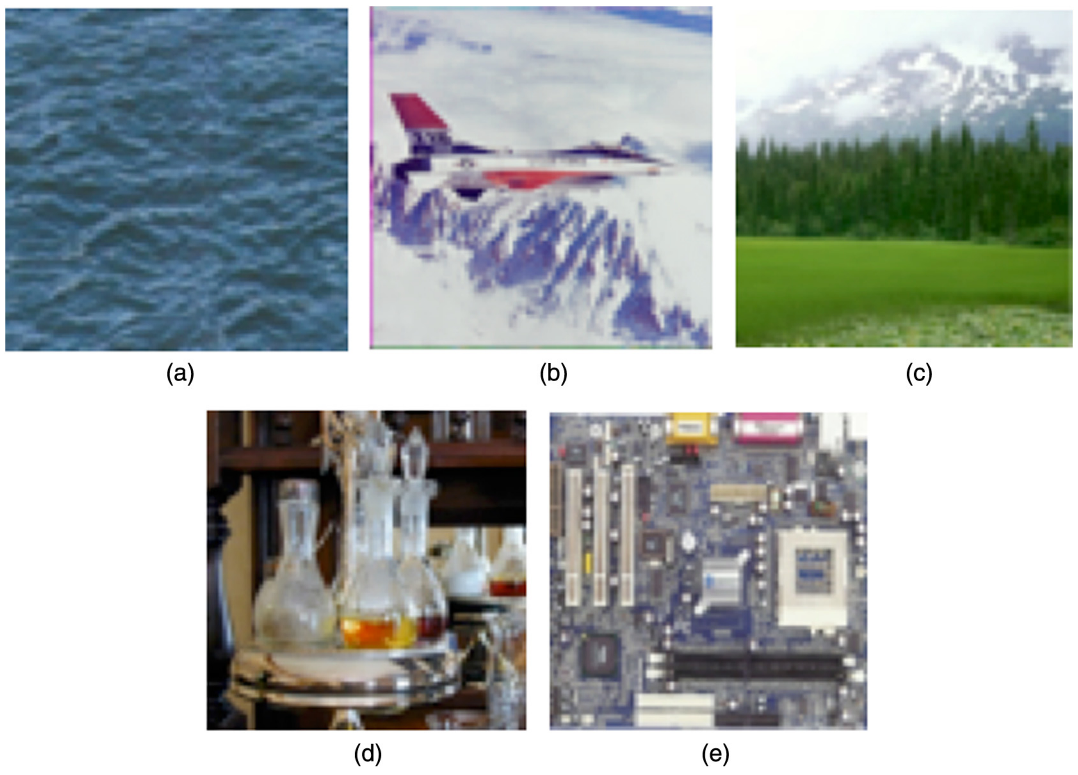


Fig. 7 Five images, (a) ocean, (b) F-16 fighter, (c) mountain, (d) caster, and (e) IC motherboard, tested to determine the threshold value for Laplacian kernel response. The image size is 512 × 512 pixels. The entropy is (a) 12.99, (b) 13.76, (c) 15.48, (d) 15.73, and (e) 16.07.

noise-free pixel. Hence, the weakness of MI can be complemented by the four 1-D Laplacian kernels.

In Fig. 6, four 3×3 1-D Laplacian kernels are used in our experiments for the second-level noise detection. In fact, both 3×3 and 5×5 1-D Laplacian kernels[4] were tested simultaneously to verify which size of the kernel is more appropriate for the second-level detection. In our experiments, the kernels of different sizes do not show any significantly different results. At the first level of noise pixel detection, the corrupted image is tested through MI index; then if necessary, four 1-D Laplacian kernels will be used in the second-level detection. Four 1-D Laplacian kernels are sensitive to edges with four directions. The minimum absolute value from the four kernel operators is chosen as the detection value.

The Laplacian kernel response is denoted as $C_{i,j}$, and $\otimes$ is a kernel operator as shown below

$$C_{i,j} = \min\{x_{i,j} \otimes K_d : d = 1 \text{ to } 4\}, \tag{15}$$

where $x_{i,j}$ is a pixel and K_d is a Laplacian kernel. Each kernel is applied to choose the minimum absolute value in the window. This value is compared with the threshold (T_0) value to determine whether it is a noise pixel or not [Eq. (16)]. Please note that we use C to represent $C_{i,j}$ for the pixel (i, j) in the examination in Eq. (16). If the value is larger than the threshold, it is an impulse noise. If the value is smaller than (or equal to) the threshold, then the current pixel is either a noise-free or edge pixel.[4] The determination of this threshold will be discussed in Sec. 4

$$f(C) = \begin{cases} C > T_0, & \text{Impulse noise} \\ C \leq T_0, & \text{Noise free or edge} \end{cases}. \tag{16}$$

From the detection based on MI and Laplacian kernel response, a pixel is either classified as a noise pixel, in which the VMF will be called in to remove the noise, or the pixel value remains unchanged.

3.3 *Proposed Filtering Process with Vector Median Filter*

The remaining issue related to the implementation of Moran's I vector median filter (MIVMF) is the determination of the threshold value because the improper threshold setting can produce unwanted filtering results. First, the threshold value ε_0 for MI should be properly set up to check if pixels in the sliding window are located at the noise area or the relatively homogenous, noise-free area.

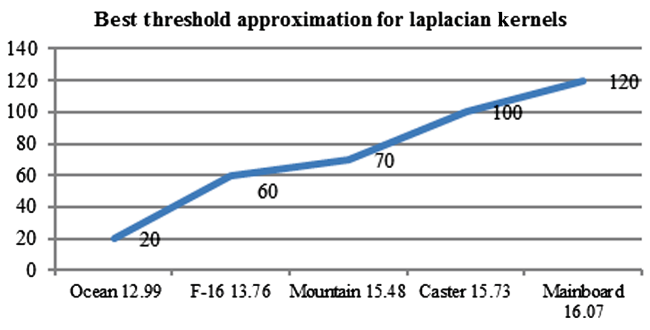


Fig. 8 A comparison chart for the threshold values used in four 1-D Laplacian kernels for images tested.

Table 9 Results on ocean image. The number next to each noise level column shows the number of iterations in the filtering process.

Threshold	5% (1)	10% (2)	15% (2)	Total
10	45.72	42.36	39.9	127.98
20	**45.74**	**42.88**	**40.18**	**128.8**
30	44.65	42.71	40.03	127.39
40	43.18	42.49	39.83	125.5
50	41.28	42.09	39.52	122.89
60	39.48	40.97	38.88	119.33
70	38.08	39.44	37.54	115.06
80	36.55	37.63	35.91	110.09
90	35.02	35.72	34.18	104.92
100	33.56	33.93	32.4	99.89
110	32.28	32.09	30.39	94.76
120	31.19	30.44	28.54	90.17
130	30.18	29	26.87	86.05
140	29.17	27.53	25.26	81.96

Note: The best threshold is highlighted with bold font.

Table 10 Results on F-16 image. The number next to each noise level column shows the number of iterations in the filtering process.

Threshold	5% (1)	10% (2)	15% (2)	Total
10	30.75	28.59	27.22	86.56
20	32.33	29.97	28.49	90.79
30	33.58	31.19	29.61	94.38
40	34.54	32.29	30.61	97.44
50	35.07	33.17	31.28	99.52
60	**34.95**	**33.76**	**31.81**	**100.52**
70	34.54	33.84	31.86	100.24
80	33.67	33.49	31.47	98.63
90	32.79	32.78	30.78	96.35
100	31.8	31.88	29.88	93.56
110	30.97	30.82	28.75	90.54
120	30.11	29.71	27.55	87.37
130	29.27	28.48	26.41	84.16
140	28.46	27.36	25.81	81.63

Note: The best threshold is highlighted with bold font.

Table 11 Results on mountain image. The number next to each noise level column shows the number of iterations in the filtering process.

Threshold	5% (1)	10% (2)	15% (2)	Total
10	28.21	26.58	25.52	80.31
20	29.53	27.73	26.58	83.84
30	30.73	28.8	27.55	87.08
40	31.73	29.79	28.37	89.89
50	32.47	30.74	29.14	92.35
60	32.98	31.61	29.73	94.32
70	**33.18**	**32.16**	**30.05**	**95.39**
80	32.98	32.31	30.08	95.37
90	32.47	32.07	29.87	94.41
100	31.77	31.52	29.31	92.6
110	31	30.72	28.62	90.34
120	30.3	29.8	27.79	87.89
130	29.6	28.69	26.75	85.04
140	28.69	27.7	25.69	82.08

Note: The best threshold is highlighted with bold font.

Table 12 Results on caster image. The number next to each noise level column shows the number of iterations in the filtering process.

Threshold	5% (1)	10% (2)	15% (2)	Total
10	23.12	21.33	20.34	64.79
20	24.05	22.01	20.92	66.98
30	24.83	22.67	21.53	69.03
40	25.53	23.3	22.05	70.88
50	26.13	23.88	22.59	72.6
60	26.58	24.38	23.05	74.01
70	26.95	24.85	23.42	75.22
80	27.26	25.22	23.69	76.17
90	27.43	25.5	23.91	76.84
100	**27.5**	**25.53**	**23.96**	**76.99**
110	27.53	25.55	23.88	76.96
120	27.35	25.3	23.57	76.22
130	27.01	24.91	23.13	75.05
140	26.58	24.4	22.58	73.56

Note: The best threshold is highlighted with bold font.

Table 13 Results on IC motherboard image. The number next to each noise level column shows the number of iterations in the filtering process.

Threshold	5% (1)	10% (2)	15% (2)	Total
10	16.18	15.24	14.65	46.07
20	16.93	15.58	14.94	47.45
30	17.59	16	15.26	48.85
40	18.21	16.45	15.63	50.29
50	18.89	16.97	16.06	51.92
60	19.6	17.55	16.56	53.71
70	20.37	18.18	17.08	55.63
80	21.19	18.86	17.64	57.69
90	21.98	19.52	18.16	59.66
100	22.72	20.16	18.57	61.45
110	23.24	20.61	18.87	62.72
120	**23.45**	**20.82**	**19**	**63.27**
130	23.51	20.8	18.91	63.22
140	23.35	20.57	18.62	62.54

Note: The best threshold is highlighted with bold font.

We chose three images as shown in Fig. 3 to verify the proper threshold value. We tested all three images with different noise levels for 5%, 10%, and 15%, and the threshold value ε_0 can be properly estimated. If MI is less than ε_0, then the Laplacian kernels will be used to determine if it is a noisy pixel. Otherwise, it is a noise-free pixel.

Algorithm 1 MIVMF.

```
Step 1: Read in a noise color image and set threshold values $\varepsilon_0$ and
        $T_0$. (Please note that the selection of threshold values $\varepsilon_0$ and
        $T_0$ was discussed in Sec. 3.3 in detail and is given in
        Table 14.)

  Steps 2 to 5 are repeated for each pixel in the image for each
      iteration.

Step 2: Calculate MI value in the neighborhood of the pixel within
        the defined filtering window for each color component of
        the color image.

Step 3: Evaluate the four 1-D Laplacian kernel response values in
        the neighborhood of the pixel and select the minimum kernel
        response value (i.e., Min) for each color component of
        the color image.

Step 4: If any MI value from all three color components is less than $\varepsilon_0$
        (first-level detection), and if the Min value from all three color
        components is larger than $T_0$ (second-level detection), go to
        step 5 (for denoising). Otherwise, go to step 2 (it is a noise-free
        pixel).

Step 5: Use the VMF for removing noise and then go to step 2.
```

Table 14 Some additional parameters for filters used in the experiments.

Filters	Extra parameters used besides 3×3 window
CWVMF	Center weight $= 4$
ACWVMF	$\lambda = 2$, $\tau = 2$, threshold $= 80$
PGVMF	$\alpha = 6$
RSVMF	$\alpha = 1.25$ for light noise, $\alpha = 1.5$ for heavy noise
Proposed	3×3 window size, Laplacian kernel threshold value, T_0 is interpolated based on the image complexity with entropy measure as shown in Fig. 8
	MI threshold $\varepsilon_0 = 0.0$

As shown in Tables 6 and 7, the best threshold value ε_0 is somewhere between -0.1 and 0.1 from the results (bold print) of PSNR and SSIM. In Table 8, the best threshold value is between -0.2 and 0. From those experimental results, this information can be used as an indicator for choosing a threshold value for MI index for a color image with different complexity in entropy. Consequently, the promising PSNR and SSIM results from Lena, peppers, and baboon images are located around $\varepsilon_0 = 0$. Hence, we set up our threshold value as $\varepsilon_0 = 0.0$ for all images in our experiments. Furthermore, the difference in the range of possible threshold values for ε_0 is so small such that the selection of this index is not critical to the filtered results.

The next threshold to be determined is the response of four 1-D Laplacian kernels. We tested many images to find out whether there exists any common threshold value which can be used for an image with different complexity. However, we could not find such a common threshold value with our initial experiments. Therefore, we set up the hypothesis that image complexity might be related

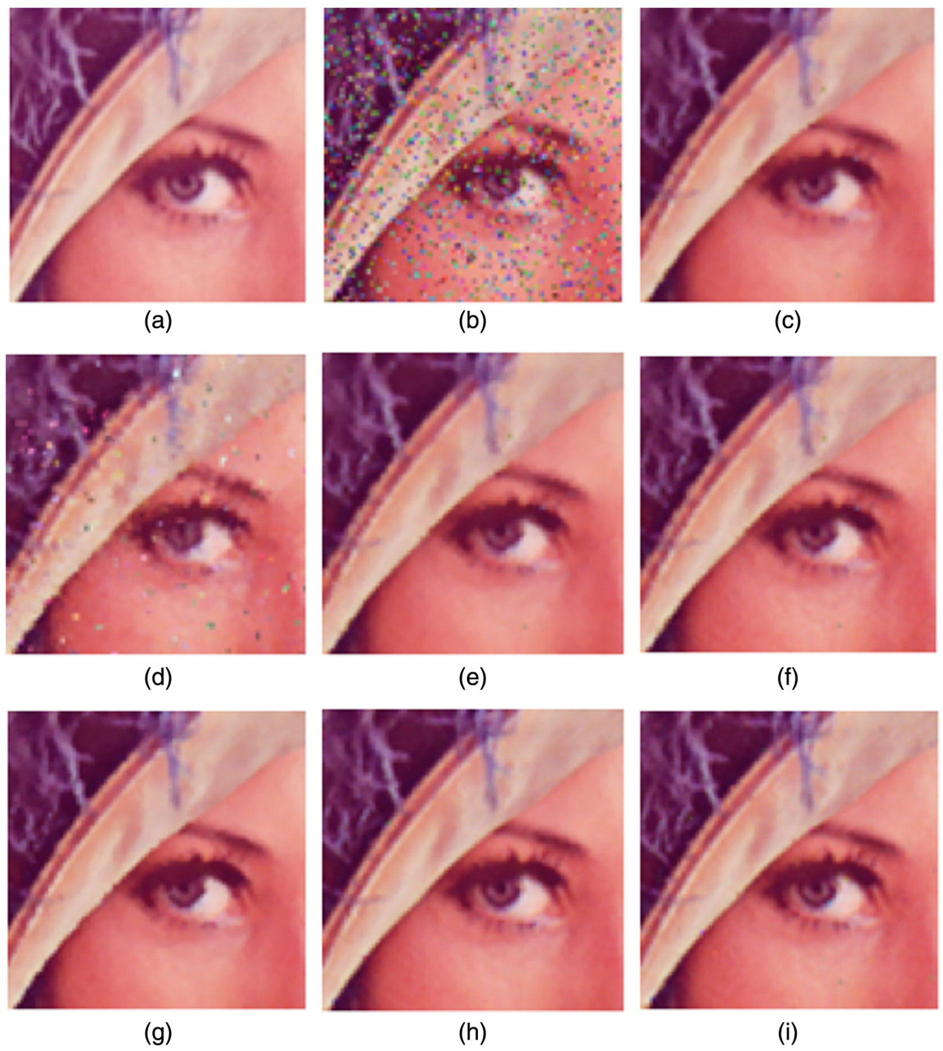


Fig. 9 Filtering results of Lena image. To see the filtering result in detail, it has been zoomed at the center area: (a) an original image, (b) 15% impulse noise added, and (c), (d), (e), (f), (g), (h), and (i) are the results of VMF, BVDF, CWVMF, ACWVMF, PGVMF, RSVMF, and MIVMF (i.e., our proposed filter), respectively.

with the threshold value of Laplacian kernels. We chose five images, as shown in Fig. 7, with different entropy levels and filtered with various threshold values. In our empirical study on the images tested, the threshold range is between 20 and 120, as shown in Fig. 8. In addition, the noise levels of 5%, 10%, and 15% were tested. In Fig. 8, only Laplacian kernels were used to determine the threshold (T_0) for the VMF. For the noise level 10% and 15%, the filtering process was run twice to improve overall outcomes measured by PSNR. For the 5% case, one filtering was good enough for removing impulse noise. The number of iterations is written next to the noise level, as shown in Tables 9–13.

By observing experimental results from Tables 9–13, we discovered a trend in the pattern the setup of the threshold value for Laplacian kernels response for an image. This trend is to increase the threshold value based on the entropy of an image being denoised, as shown in Fig. 8. Therefore, it is quite straightforward to calculate the entropy of an image to estimate the threshold value. Based on this assumption, we used the interpolation method for assessing a threshold value for an image being denoised in our simulation in Sec. 4.

From the discussions above, our proposed denoising Algorithm 1 (MIVMF) is summarized in the steps. Please note that the algorithm can be repeated if needed. In other words, the output from the first pass of the algorithm becomes the input for the second pass and so on.

4 Experimental Results

To evaluate the performance of the proposed algorithm and compare it with other well-developed denoising filters, we tested all the filters listed in Table 4. Various test parameters, including the filtering window size and threshold values, are listed in Table 14. All test conditions except our proposed filter are based on the suggested parameter values from the literature reviewed. The size of the neighborhood window used for all filters is 3×3. To compare the denoising performance, PSNR, SSIM, and time complexity measured in seconds are calculated. Three images from Fig. 3 are used in the experiments. Some of denoised images are shown in Figs. 9–12. Please note that in the noise levels 10% and 15%, both RSVMF and MIVMF were run twice to obtain the best outcome measured by PSNR. Other filters were run only

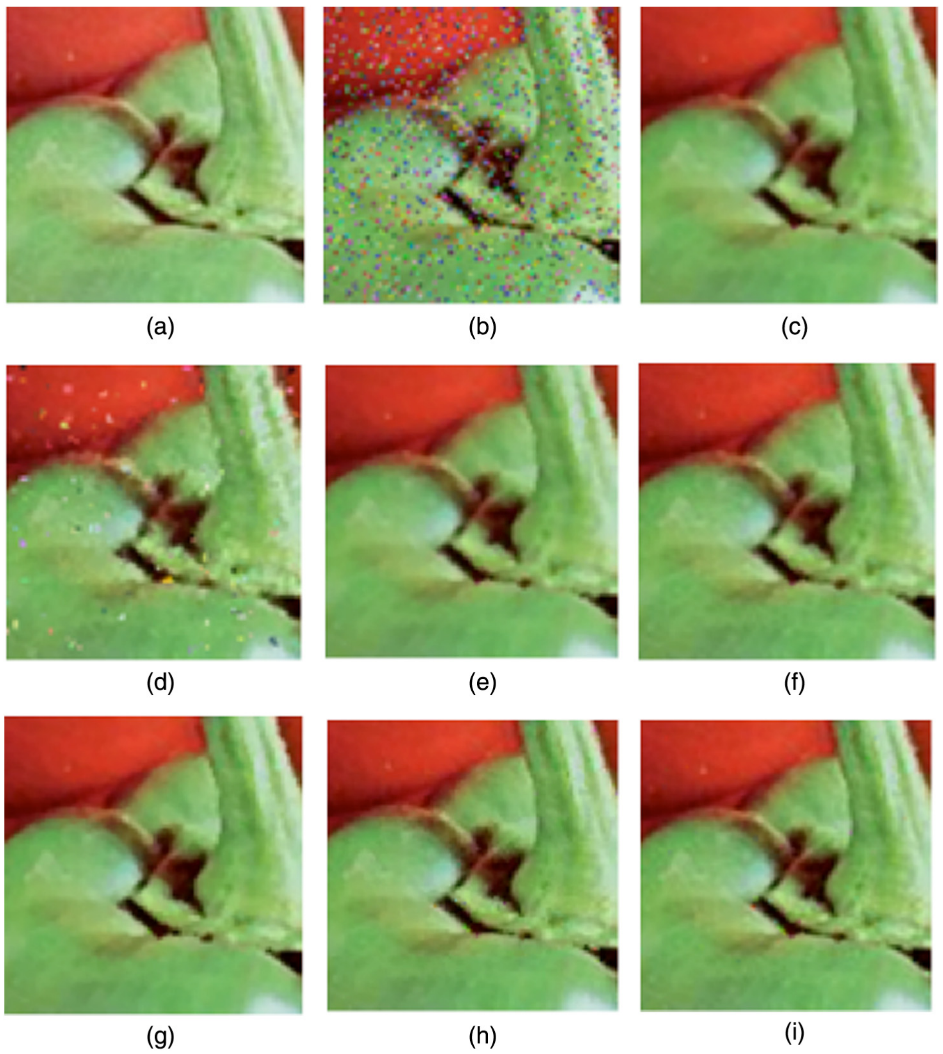


Fig. 10 Filtering results of peppers image. To see the filtering result in detail, it has been zoomed at the center area: (a) an original image, (b) 15% impulse noise added, and (c), (d), (e), (f), (g), (h), and (i) are the results of VMF, BVDF, CWVMF, ACWVMF, PGVMF, RSVMF, and MIVMF (i.e., our proposed filter), respectively.

once. (Otherwise, the image will be blurred if they were run more than once.) For the repetition, the output from the first pass of the algorithm becomes the input for the second pass and so on.

In our experiments, there is no significant difference for a filter in terms of the time complexity for processing an image with any noise level or image complexity based on the entropy measure. This can be seen from Tables 15–17. The proposed MIVMF is the best among all the filters tested with the PSNR measure. For the SSIM measure, which is for human eye perception, all the filters are almost identical, except that BVDF is degrading when the noise level is increased.

On the contrary, for an image with medium entropy, such as the peppers image, the PSNR measure for all filters is decreasing when the noise level is increased, as shown in Table 16. This phenomenon is also shown in the SSIM measure for the BVDF. Otherwise, the SSIM is very similar for all the filters tested. The proposed MIVMF has the best PSNR measure for an image with low, medium, and high entropy. For images with higher entropy, a trend exists for all filters that the SSIM is decreasing when the noise level is increasing, except for the VMF in which the 15% noise shows a high SSIM value, as shown in Table 17. In terms of the PSNR measure, all of the filters tested decrease if the image complexity is increased. However, the MIVMF has the overall highest PSNR value at different noise levels and with almost any complexity of images measured in entropy.

Based on the results shown in Tables 15–17, there is only slight difference on the performance of RSVMF and MIVMF for the low noise level. Hence, to measure the effectiveness of RSVMF and MIVMF for heavy noises, both filters were tested on 20%, 25%, and 30% noise levels on three images, and the results are shown in Tables 18–20. (Please note that the results are from three iterations of each filter, and overall time spent is the sum of three iterations.) Test threshold (α) for RSVMF is 1.5 based on Ref. 12. While the SSIM is similar for both filters, the proposed MIVMF has a better PSNR measure with the higher noise level. The MIVMF has the

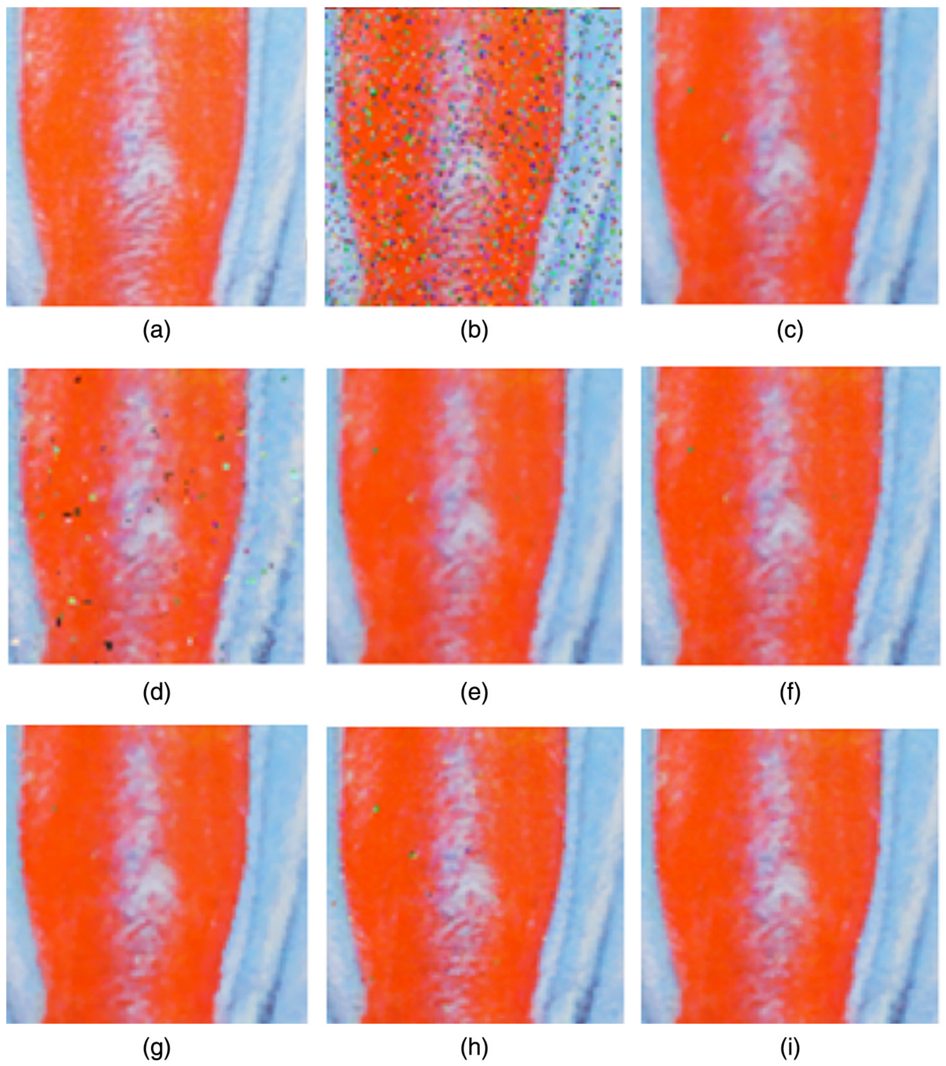
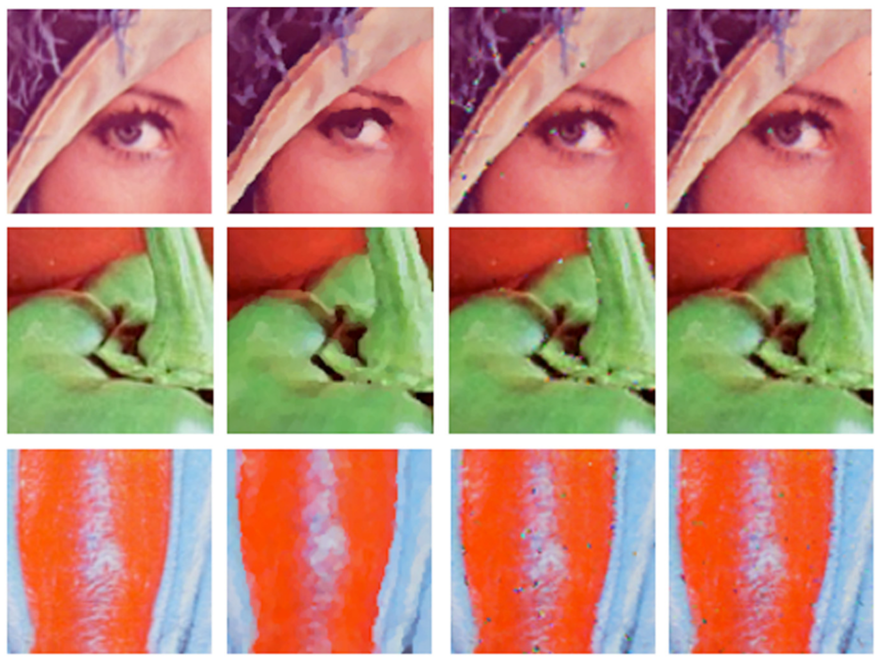
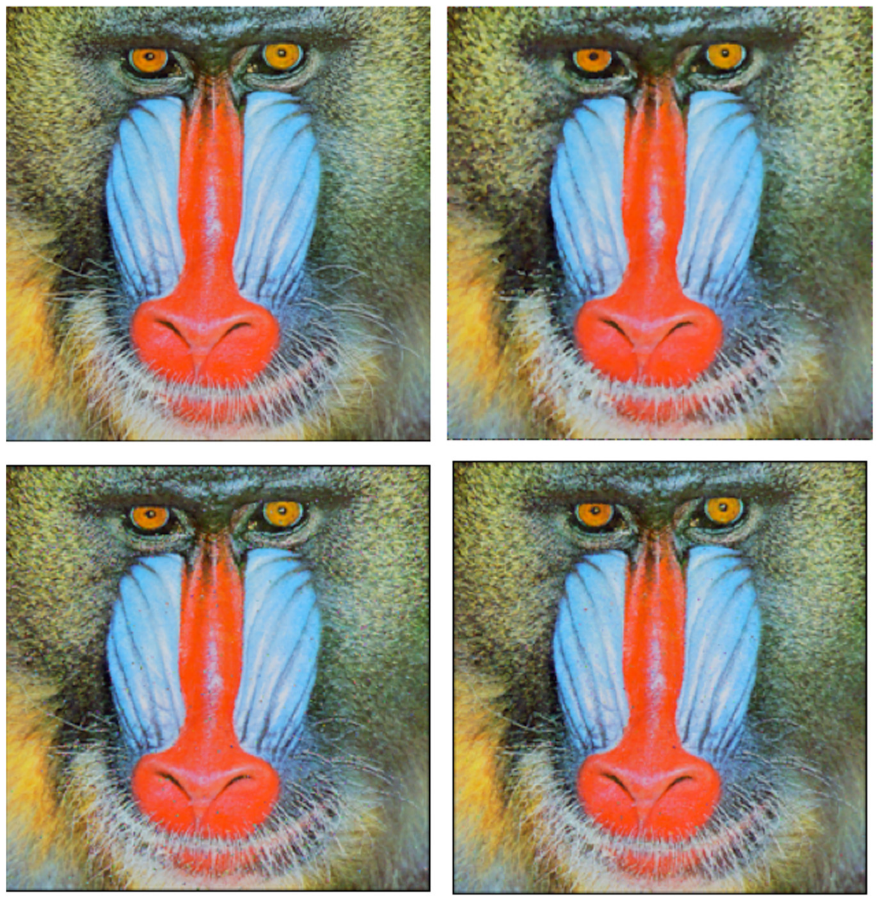


Fig. 11 Filtering results of baboon image. To see the filtering result in detail, it has been zoomed at the center area: (a) an original image, (b) 15% impulse noise added, and (c), (d), (e), (f), (g), (h), and (i) are the results of VMF, BVDF, CWVMF, ACWVMF, PGVMF, RSVMF, and MIVMF (i.e., our proposed filter), respectively.



(a)



(b)

Fig. 12 (a) Specific filtering results of 30% noise added cases for the comparison among PGVMF, RSVMF, and MIVMF (i.e., our proposed filter). To see the filtering result in detail, it has been zoomed at the center area. Three original images are shown in the first column, the second column are results of PGVMF, the third column are results of RSVMF, and the last column are results of MIVMF. (b) The entire baboon image is shown to illustrate the denoising effect of PGVMF, RSVMF, and MIVMF. We can clearly see that the filtered result by PGVMF is distorted; in particular, the moustache of the baboon became very thick. Each filter was repeated three times to obtain the result.

Table 15 Tested results of Lena with entropy 15.12 and $T_0 = 67$ in 5%, 10%, and 15% noise levels.

Time spent	VMF (s)	BVDF (s)	CWVMF (s)	ACWVMF (s)	PGVMF (s)	RSVMF (s)	MIVMF (s)
5%	3.67	7.79	3.68	11.72	4.44	3.88	**1.26**
10%	3.68	7.64	3.70	11.64	4.48	7.72	**2.53**
15%	3.69	7.65	3.70	11.79	4.48	7.76	**2.78**
PSNR	VMF	BVDF	CWVMF	ACWVMF	PGVMF	RSVMF	MIVMF
5%	28.34	24.92	27.60	29.94	26.99	34.52	**34.85**
10%	27.81	22.59	27.12	29.06	26.89	32.40	**32.99**
15%	27.19	20.41	26.58	28.16	26.75	30.11	**30.93**
SSIM	VMF	BVDF	CWVMF	ACWVMF	PGVMF	RSVMF	MIVMF
5%	0.9824	0.9515	0.9806	0.9907	0.9782	0.9947	**0.9971**
10%	0.9799	0.9061	0.9780	0.9879	0.9773	0.9898	**0.9954**
15%	0.9769	0.8524	0.9750	0.9849	0.9772	0.9881	**0.9926**

Note: In 10% noise, the MIVMF was run two iterations, which required 1.48 and 1.05 s (2.53-s total) for first and second iteration, respectively. Similarly, for 15%, it took 1.71 and 1.07 s (2.78-s total). For RSVMF, we also ran two iterations for 10% and 15%. For 10% noise, it took 3.87 and 3.85 s (7.72-s total). For 15%, it took 3.96 and 3.80 s (7.76-s total). s denotes seconds.

Table 16 Test results of peppers with entropy 15.30 and $T_0 = 68$ in 5%, 10%, and 15% noise levels.

Time spent	VMF (s)	BVDF (s)	CWVMF (s)	ACWVMF (s)	PGVMF (s)	RSVMF (s)	MIVMF (s)
5%	3.53	7.66	3.53	10.76	4.32	3.70	**1.19**
10%	3.55	7.60	3.55	11.07	4.33	7.36	**2.38**
15%	3.57	7.63	3.57	11.42	4.43	7.39	**2.65**
PSNR	VMF	BVDF	CWVMF	ACWVMF	PGVMF	RSVMF	MIVMF
5%	31.44	26.19	30.12	32.03	29.03	**35.76**	35.50
10%	30.39	23.00	29.34	30.84	28.91	33.78	**33.88**
15%	29.11	19.97	28.31	29.42	28.62	31.28	**31.84**
SSIM	VMF	BVDF	CWVMF	ACWVMF	PGVMF	RSVMF	MIVMF
5%	0.9891	0.9583	0.9878	0.9955	0.9853	0.9970	**0.9974**
10%	0.9871	0.9058	0.9858	0.9937	0.9851	0.9939	**0.9969**
15%	0.9837	0.8228	0.9822	0.9901	0.9844	0.9927	**0.9947**

Note: Similar to the note in Table 15: In 10%, the MIVMF took 1.43 and 0.95 s (2.38-s total) for first and second iteration, respectively. For 15%, it spent 1.66 and 0.99 s (2.65-s total). For RSVMF with 10%, it took 3.73 and 3.63 s (7.36-s total). For 15%, it took 3.75 and 3.64 s (7.39-s total). s denotes seconds.

Table 17 Test results of baboon with entropy 15.90 and $T_0 = 109$ in 5%, 10%, and 15% noise levels.

Time spent	VMF (s)	BVDF (s)	CWVMF (s)	ACWVMF (s)	PGVMF (s)	RSVMF (s)	MIVMF (s)
5%	3.81	7.79	3.68	12.93	4.49	3.98	**1.25**
10%	3.72	7.65	3.77	13.11	4.49	7.77	**2.47**
15%	3.73	7.65	3.72	13.19	4.47	7.76	**2.70**
PSNR	VMF	BVDF	CWVMF	ACWVMF	PGVMF	RSVMF	MIVMF
5%	17.98	14.88	17.70	17.92	17.29	24.29	**25.77**
10%	17.76	14.53	17.48	17.68	17.24	21.91	**23.30**
15%	17.52	14.12	17.24	17.41	17.14	20.49	**21.70**
SSIM	VMF	BVDF	CWVMF	ACWVMF	PGVMF	RSVMF	MIVMF
5%	0.9144	0.8125	0.9103	0.9171	0.8993	0.9823	**0.9876**
10%	0.9055	0.7900	0.9012	0.9078	0.8968	0.9664	**0.9767**
15%	0.9871	0.7598	0.8925	0.8988	0.8937	0.9563	**0.9639**

Note: Similar to the note in Table 15: In 10% noise, our proposed MIVMF took 1.42 and 1.05 s (2.47-s total) for first and second iteration, respectively. For 15%, it spent 1.60 and 1.10 s (2.70-s total). For RSVMF with 10%, it took 3.89 and 3.89 s (7.77-s total). For 15%, it took 3.90 and 3.86 s (7.76-s total). s denotes seconds.

Table 18 A comparison of RSVMF and MIVMF on heavy noise added Lena image.

Time spent	RSVMF (s)	MIVMF (s)
20% noise	11.37	**4.20**
25% noise	11.37	**4.48**
30% noise	11.35	**4.71**
PSNR	RSVMF	MIVMF
20% noise	28.34	**29.46**
25% noise	26.72	**28.22**
30% noise	24.37	**26.85**
SSIM	RSVMF	MIVMF
20% noise	0.9869	**0.9881**
25% noise	0.9802	**0.9828**
30% noise	0.9649	**0.9732**

Note: s denotes seconds.

Table 19 A comparison of RSVMF and MIVMF on heavy noise added peppers image.

Time spent	RSVMF (s)	MIVMF (s)
20% noise	10.98	**3.99**
25% noise	10.95	**4.26**
30% noise	11.05	**4.58**
PSNR	RSVMF	MIVMF
20% noise	29.40	**30.27**
25% noise	26.75	**28.70**
30% noise	24.48	**27.31**
SSIM	RSVMF	MIVMF
20% noise	0.9896	**0.9916**
25% noise	0.9859	**0.9862**
30% noise	0.9707	**0.9788**

Note: s denotes seconds.

Table 20 A comparison of RSVMF and MIVMF on heavy noise added baboon image.

Time spent	RSVMF (s)	MIVMF (s)
20% noise	11.51	**4.11**
25% noise	11.57	**4.32**
30% noise	11.58	**4.55**
PSNR	RSVMF	MIVMF
20% noise	19.18	**20.47**
25% noise	17.98	**19.49**
30% noise	16.66	**18.49**
SSIM	RSVMF	MIVMF
20% noise	0.9402	**0.9496**
25% noise	0.9217	**0.9328**
30% noise	0.8984	**0.9104**

Note: s denotes seconds.

overall highest PSNR value in those noise levels. In addition, the running time for the MIVMF is less than half of the RSVMF.

To further demonstrate the effectiveness of the proposed algorithm in denoising, we tested on some different images and performed a number of iterations for each filter listed in Table 14. Images tested are shown in Fig. 13 which includes parrot (entropy 15.25), penguin (entropy 15.40), and koala images (entropy 16.18). As mentioned in Table 14, we used 0.0 for MI threshold ε_0 and threshold T_0 is interpolated based on the image entropy. Table 21 shows the PSNR score for each filter tested. To each image 5% noise was added, and each filter was run only once in the experiments. Experimental results show that the proposed algorithm, MIVMF, has the best PSNR scores.

In our study, some filters will cause the distortion of images and blur the image details if the filter is run multiple times on the image. To test the efficiency of the proposed algorithm, we ran each filter with a number of iterations to compare the filtered results. The results were obtained with six iterations for each filter and are shown in Table 21. All PSNR scores for VMF, BVDF, CWVMF, ACWVMF, PGVMF, and RSVMF are decreasing when the number of iterations is increased. This phenomenon will continue if the filter is iterated again. This indicates that the filtered image is distorted and image details are not preserved. The PSNR scores with our proposed MIVMF either decrease or increase slightly, as shown in Table 21. In either case, the MIVMF will reach a saturation state, and its PSNR score will not change again. For example, in Tables 21(a) and 21(d) (first column), the PSNR of the MIVMF will stay in 23.57 and 26.43, respectively, no matter how many more iterations are repeated. Similarly, in Tables 21(b) and 21(c) (first column), the PSNR will stay in 29.59 and 30.33, respectively. This important characteristic shows that the MIVMF is an efficient filter with the good detection mechanism for determining if a pixel is noise or not. Hence, the image details will not be damaged or blurred. We also observed that the MIVMF has the highest PSNR value no matter how many iterations were repeated. In addition, the MIVMF runs much faster than other filters, as shown in Tables 15–20.

By observing the experimental results in Tables 15–21 and Figs. 9–12, the proposed MIVMF shows a promising result compared with other vector-based filters. We developed an efficient denoising filter for accomplishing better time complexity and efficient denoising results, which was the goal set up at the beginning of this study. Our two-level detection function based on MI and 1-D Laplacian kernels works properly, so it can reduce the processing time and produce better filtered results compared with other similar type of vector-based filters.

5 Conclusion and Future Work

We propose a noise detection and removal filter that uses MI statistics (i.e., MI value) to calculate the spatial autocorrelation indices and combine with four 1-D Laplacian kernels for random-valued impulse noise detection for color images. As far as we know, this may be the first attempt to use MI statistics for color image denoising. This is an adaptive filter in which the MI value and response of four 1-D Laplacian kernels are calculated for each 3×3 neighborhood window. This two-level noise detection scheme works efficiently as it has the capability of distinguishing a given pixel as noise or not. The VMF is then used to process those noisy pixels and leave noise-free pixels untouched. Our proposed filter can be considered an efficient extension of the VMF in a sense. As an extension, the proposed filter can achieve

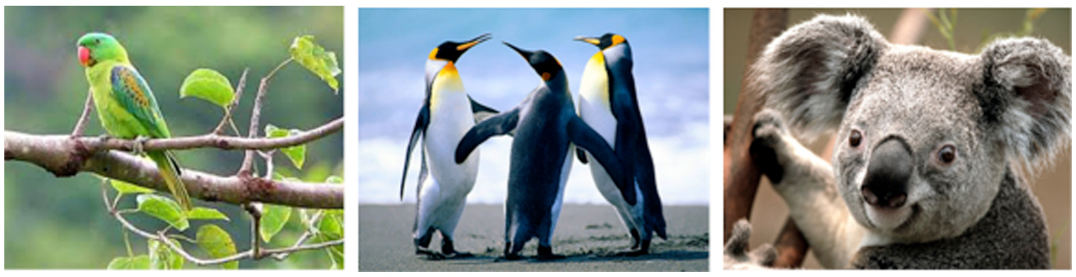

(a) (b) (c)

Fig. 13 Three images, (a) parrot with 255 × 197 pixels, (b) penguin with 1024 × 768 pixels, and (c) koala with 1024 × 768 pixels. The entropy for (a) is 15.25, (b) 15.40, and (c) 16.18. Using the interpolation method shown in Fig. 8, threshold value of T_0 is (a) is 79, (b) 78, and (c) 120. Please note that we used 0.0 for ε_0 for all the experiments in this paper.

Table 21 The PSNRs for different numbers of iterations and filters are listed: (a) parrot was corrupted with 5% noise, (b) penguin with 5% noise, (c) koala with 5% noise, and (d) parrot with 15% noise. Each filter was run six times in the experiments and is represented with the respective number in the first column, and then each PSNR was calculated for each run. The saturated modes were highlighted with italicized font in the table for the MIVMF filter.

No. of iterations	VMF	BVDF	CWVMF	ACWVMF	PGVMF	RSVMF	MIVMF
(a) 1	19.75	16.50	19.46	19.61	19.08	22.79	**24.18**
(a) 2	19.28	16.38	19.14	19.29	18.24	21.72	**23.72**
(a) 3	18.86	15.94	18.64	18.86	17.71	21.34	**23.60**
(a) 4	18.62	15.60	18.40	18.66	17.46	21.18	**23.58**
(a) 5	18.45	15.27	18.20	18.48	17.29	21.10	***23.57***
(a) 6	18.33	15.06	18.08	18.39	17.18	21.07	***23.57***
(b) 1	24.94	20.48	24.40	24.96	23.56	28.47	**29.49**
(b) 2	24.10	19.84	23.65	24.21	22.30	27.30	**29.66**
(b) 3	23.55	19.28	23.05	23.71	21.65	26.88	**29.61**
(b) 4	23.24	18.91	22.72	23.44	21.34	26.70	***29.59***
(b) 5	23.05	18.64	22.50	23.26	21.16	26.61	***29.59***
(b) 6	22.91	18.41	22.36	23.15	21.05	26.56	***29.59***
(c) 1	24.58	19.63	24.27	24.92	23.59	28.68	**29.13**
(c) 2	23.15	18.97	23.03	23.79	22.16	27.39	**30.31**
(c) 3	22.31	18.28	22.16	23.09	21.26	26.93	***30.33***
(c) 4	21.83	17.80	21.67	22.70	20.79	26.75	***30.33***
(c) 5	21.52	17.45	21.34	22.46	20.51	26.66	***30.33***
(c) 6	21.32	17.18	21.14	22.30	20.34	26.63	***30.33***
(d) 1	18.71	14.66	18.45	18.56	18.39	20.43	**27.91**
(d) 2	18.46	15.27	18.33	18.42	18.24	20.28	**26.76**
(d) 3	18.07	15.14	17.91	18.05	17.27	20.01	**26.51**
(d) 4	17.85	14.92	17.68	17.85	17.01	19.89	**26.44**
(d) 5	17.70	14.76	17.51	17.73	16.83	19.82	***26.43***
(d) 6	17.60	14.54	17.41	17.63	16.73	19.78	***26.43***

Note: The saturated modes were highlighted with italicized font for the MIVMF filter.

better denoising results and efficient time complexity for the color images, which are corrupted with random-valued impulse noises. To determine if a pixel is noise or not, two levels of detection are used. The first level is controlled by the MI value, and the second level is measured by the response of four 1-D Laplacian kernels. An empirical study was able to estimate the threshold value for MI statistics. An interpolation method, which can roughly estimate the threshold value that is used in the comparison with the response of four 1-D Laplacian kernels for an image being denoised based on the entropy measure, was also established. The threshold value calculated with this interpolation method was tested in our experiments and illustrated with good denoising results.

MI statistics shows an interesting result in our exploration of the application of color images for detecting random-valued impulse noises. We compared the proposed MIVMF with other vector-type MFs, including VMF, BVDF, CWVMF, ACWVMF, PGVMF, and RSVMF, using the criteria of PSNR and SSIM. The proposed MIVMF shows promising denoising results based on these criteria. The MIVMF is faster than the other filters tested in the experiments regardless of the noise level and image complexity. For the PSNR, the MIVMF has a higher value than the compared filters. By the visualization, the proposed MIVMF can avoid the image blurring results, preserve edge details in the image, and achieve superior noise reduction. We found that this kind of spatial statistic is very useful in the color image denoising algorithm.

Although we were able to estimate the threshold values that were successfully used in our proposed filter, the issue of determining an optimal and adaptive threshold value for detecting a noise pixel remains a challenging problem in our exploration of MI statistics. If this threshold value can be estimated with an adaptive approach, it would be more useful to generate better denoising outcomes. We only applied the concept of spatial autocorrelation based on MI statistics and 1-D Laplacian kernels for the impulse noise detection in color images. We concentrated on the nonfuzzy vector-type MFs in this work. Several fuzzy vector-type filters that work well for noise detection and removal also exist. For future work, we plan to investigate the following four tasks: (1) to develop an adaptive method that can estimate the threshold values more accurately, (2) to determine if the MIVMF is suitable and efficient for hyperspectral remote sensing images, (3) to extend our detection scheme to other noise models, such as Gaussian noise, and (4) to explore the fuzzy vector-type MFs and determine if the fuzzy mathematics can be used to improve the proposed MIVMF performance for denoising.

Acknowledgments

The authors would like to thank all the reviewers for their constructive comments, which helped us to improve the quality of the paper.

References

1. M. Lebrun et al., "Secrets of image denoising cuisine," *Acta Numer.* **21**, 475–576 (2012).
2. X. L. Xie and G. Beni, "A validity measure for fuzzy clustering," *IEEE Trans. Pattern Anal. Mach. Intell.* **13**(8), 841–847 (1991).
3. Z. Zhou, "Cognition and removal of impulse noise with uncertainty," *IEEE Trans. Image Process.* **21**(7), 3157–3167 (2012).
4. S. Zhang and M. A. Karim, "A new impulse detector for switching median filters," *IEEE Signal Process. Lett.* **9**(11), 360–363 (2002).
5. H.-L. Eng and K.-K. Ma, "Noise adaptive soft-switching median filter," *IEEE Trans. Image Process.* **10**(2), 242–251 (2001).
6. E. Abreu et al., "A new efficient approach for the removal of impulse noise from highly corrupted images," *IEEE Trans. Image Process.* **5**(6), 1012–1025 (1996).
7. F. Ahmed and S. Das, "Removal of high-density salt-and-pepper noise in images with an iterative adaptive fuzzy filter using alpha-trimmed mean," *IEEE Trans. Fuzzy Syst.* **22**(5), 1352–1358 (2014).
8. R. Lukac and K. N. Plataniotis, *Color Image Processing, Method and Applications*, CRC Press, Boca Raton, Florida (2007).
9. V. Novoselac and B. Zovko-Cihlar, "Image noise deduction by vector median filter," in *54th Int. Symp. ELMAR-2012*, Zadar, Croatia (2012).
10. M. E. Celebi, H. A. Kingravi, and Y. A. Aslandogan, "Nonlinear vector filtering for impulse noise removal from color images," *J. Electron. Imaging* **16**(3), 033008 (2007).
11. B. Smolka, K. Plataniotis, and A. N. Venetsanopoulos, "Nonlinear techniques for color image processing," in *Nonlinear Signal and Image Processing: Theory, Methods, and Applications*, K. E. Barner and G. R. Arce, Eds., pp. 445–505, CRC Press, Boca Raton, Florida (2004).
12. M. E. Celebi and Y. A. Aslandogan, "Robust switching vector median filter for impulsive noise removal," *J. Electron. Imaging* **17**(4), 043006 (2008).
13. R. Lukac et al., "Vector sigma filters for noise detection and removal in color images," *J. Visual Commun. Image Represent.* **17**, 1–26 (2006).
14. T.-J. Chen et al., "A novel image quality index using Moran *I* statistics," *Phys. Med. Biol.* **48**(8), N131 (2003).
15. K.-S. Chuang and H.-K. Huang, "Assessment of noise in a digital image using the join-count statistic and the Moran test," *Phys. Med. Biol.* **37**(2), 357–369 (1992).
16. J. Astola, P. Haavisto, and Y. Neuvo, "Vector median filters," *Proc. IEEE* **78**(4), 678–689 (1990).
17. S. Liu, "Adaptive scalar and vector median filtering of noisy color images based on noise estimation," *IET Image Process.* **5**(6), 541–553 (2010).
18. R. Lukac and B. Smolka, "Application of the adaptive center-weighted vector median framework for the enhancement of cDNA microarray images," *Int. J. Appl. Math. Comput. Sci.* **13**(3), 369–383 (2003).
19. A. Nagabhushna, "Quality assessment of resultant images after processing," *Comput. Eng. Intell. Syst.* **3**(7), 105–112 (2012).
20. W. Malpica and A. Bovik, "SSIM based range image quality assessment," in *4th Int. Workshop on Video Processing and Quality Metrics for Consumer Electronics*, Scottsdale, Arizona (2009).
21. B. Smolka and A. Chydzinski, "Fast detection and impulsive noise removal in color images," *Real-Time Imaging* **11**, 389–402 (2005).
22. R. Lukac et al., "Vector filtering for color imaging, open a world of possibility," *IEEE Signal Process. Mag.* **22**(1), 74–86 (2005).
23. T. Chen and H. Wu, "Adaptive impulse detection using center-weighted median filter," *IEEE Signal Process. Lett.* **8**(1), 1–3 (2001).
24. C. Kenny et al., "Peer group image enhancement," *IEEE Trans. Image Process.* **10**(2), 326–334 (2001).
25. B. Smolka and M. Perczak, "Generalized vector median filter," in *Proc. of the 5th Int. Symp. on Image and Signal Processing and Analysis* (2007).
26. B. Smolka, "Soft switching technique for impulsive noise removal in color images," in *5th Int. Conf. on Computational Intelligence, Communication Systems and Networks* (2013).
27. E. Chang et al., "A denoising algorithm for remote sensing images with impulse noise," in *IEEE Int. Conf. on Geoscience and Remote Sensing (IGARSS16)*, Beijing, China (2016).
28. P. Rogerson, *Statistical Methods for Geography*, 2nd ed., Sage Publications, London (2006).
29. P. Moran, "The interpretation of statistical maps," *J. R. Stat. Soc. Ser. B* **10**, 243–251 (1948).

Chih-Cheng Hung is a professor of computer science at Kennesaw State University (KSU), Georgia, USA. He was with the Intergraph Corporation from 1990 to 1993 for research and development of remote sensing image processing software. He was also with the Information Processing Lab in Anyang Normal University, China, and the Laboratory for Machine Vision and Security Research in KSU. His research interests include image processing, pattern recognition, and artificial intelligence.

Eun Suk Chang received his BS degree in geoeducation from Korea University in Seoul, South Korea, in 2003 and his MS degree in computer science from KSU, Georgia, in 2015. Currently, he works as a Windows developer at RenewedVision.com. He was with the Laboratory for Machine Vision and Security Research in KSU. His research interests include geography information science, remote sensing, image processing, spatial statistics, and data mining.