Inverse Problems and Imaging
Volume 13, No. 5, 2019, 903–930
doi:10.3934/ipi.2019041

# PIECEWISE CONSTANT SIGNAL AND IMAGE DENOISING USING A SELECTIVE AVERAGING METHOD WITH MULTIPLE NEIGHBORS

Weina Wang

Department of Mathematics, Hangzhou Dianzi University
Hangzhou 310018, China

Chunlin Wu* and Jiansong Deng

School of Mathematical Sciences, Nankai University
Tianjin 300071, China
School of Mathematical Sciences, University of Science and Technology of China
Hefei 230026, China

(Communicated by Hui Ji)

Abstract. Piecewise constant signals and images are an important kind of data. Typical examples include bar code signals, logos, cartoons, QR codes (Quick Response codes), and text images, which are widely used in both general commercial and automotive industry use. One previous work called a general selective averaging method (GSAM) was introduced to remove noise from them. It chooses homogeneous neighbors from the two closest pixels (one pixel at each side) to update the current pixel. One limitation is that it suffered from appearing sparse noisy pixels in the denoised result when the noise level is high. In this paper, we try to solve this problem by proposing a selective averaging method with multiple neighbors. To update the intensity value at each pixel, the proposed algorithm averages more homogeneous neighbors selected from a large domain, which is based on the property of the local geometry of signals and images. This greatly reduces sparse noisy pixels left in the final result by GSAM. Similarly, our method adopts the Neumann boundary condition at edges, and thus preserves edges well. In 1D case, some theoretical results are given to guarantee the convergence of our algorithm. In 2D case, except eliminating additive Gaussian noise, this algorithm can be used for restoring noisy images corrupted by speckle noise. Intensive experiments on both gray and color image denoising demonstrate that the proposed method is quite effective for piecewise constant image denoising and achieves superior performance visually and quantitatively.

1. **Introduction.** Piecewise constant signals and images are sampled from piecewise constant functions, which are an important kind of data and widely used in both commercial and automotive industry use. Typical examples include bar code signals, logos, cartoons, QR codes [34] (Quick Response codes) and text images.

2010 *Mathematics Subject Classification.* Primary: 68U10.

*Key words and phrases.* Selective averaging, multiple neighbors, piecewise constant, additive Gaussian noise, speckle noise.

The first author is supported by HDU grant KYS075618129. The second author is supported by NSFC grant 11871035 and 11531013 and Recruitment Program of Global Young Expert. The third author is supported by NSFC grant 11771420.

* Corresponding author: Chunlin Wu.

 

Each of them consists of some flat-regions separated by distinct edges. Those flat-parts are called homogeneous regions, where each is a connected component with the same intensity values. Due to the imperfection of imaging devices, signals and images are unavoidable to be degraded by different kinds of noise. Gaussian and speckle noise are two important types of noise in digital and ultrasound images, respectively. In this paper, we focus on the problem of removing Gaussian noise from piecewise constant signals and images, and make a preliminary exploration to eliminate speckle noise from this kind of images.

Image denoising is an active research area in image processing, where the recovered image is for direct visualization or further image analysis. Its main is to eliminate noise while preserving edges and features of an image. Many techniques have been proposed and studied for Gaussian noise removal from natural images, e.g., [22], [35], [42], [18], [19], [4], [14], [37], [8], [21], [5], [20], [47], [15], [16], [24], [7], [49], [38], [6], [17], [27], [3]. In particular, some algorithms also were proposed for binary image denoising in the literature [2], [9], [11], [23], [39]. There also have been some works to remove speckle noise from medical ultrasound images, e.g., [31], [30], [29], [33], [13], [40], [48], [26], [32], [28], [12], [50], [43]. These methods achieved promising denoising results for natural, binary and ultrasound images. However, there may be room for improvements by exploiting special structures of piecewise constant images.

In [44], a general selective averaging method (GSAM) was presented to estimate Gaussian noise from piecewise constant signals and images. In each iteration, the GSAM chose homogeneous pixels from 1-neighborhood of the current pixel, and updated its value by averaging these selected pixels with flexible weights. This method can be considered as a discrete approximation of the linear diffusion in flat-regions, while a discrete approximation of the linear diffusion with the Neumann boundary condition [41] at edges. It prevents the diffusion between different flat-regions, and thus image edges are preserved very well. However, it led to sparse noisy pixels appearing in the output, when dealing with images with high level noise. We need additional steps as proposed in [45] to detect and suppress them. Furthermore, since the GSAM only selects pixels from 1-neighborhood for averaging, it usually needs too many iterations to obtain satisfactory results in practical application.

Motivated by GSAM for preventing the diffusion between different flat-regions, we propose a novel algorithm to select more pixels from a larger homogeneous region by considering the property of the local geometry of signals and images. The proposed algorithm is able to not only improve the efficiency of removing noise, but also there is only very few number of noisy pixels left in the restored result. In 1D case, we have some theoretical results to guarantee the convergence of our method. Moreover, we observe that the proposed algorithm has better ability of eliminating noise than GSAM [44] for siganls with the same noise level. Compared to other existing methods, experimental results show that our method achieves superior performance for piecewise constant images with Gaussian or speckle noise.

The rest of this paper is organized as follows. In section 2, we briefly review the GSAM [44]. Then, we introduce the selective averaging with multiple neighbors method (SAMNM) in 1D case and provide some theoretical results. In section 3, we present the alternating SAMNM in 2D case. Section 4 shows experiments and comparisons for both gray and color images corrupted by additive Gaussian or speckle noise. Finally, some conclusions are summarized in section 5.

2. **The proposed algorithm in 1D case.** In this section, we first give a brief review of the 1D general selective averaging method (GSAM) [44], and then present the novel selective algorithm in detail.

Let $u \in \mathbb{R}^m$ be the ground truth, which is a discrete signal consisting of several constant segments. Each constant segment is a homogeneous region. The observed signal $f$ is corrupted by a zero-mean Gaussian noise with the standard deviation $\sigma$. We denote the index set as $\mathcal{A} = \{1, 2, \cdots, m\}$.

2.1. **The general selective averaging method for 1D signal.** Since the general selective averaging method (GSAM) [44] is an iterative algorithm, a signal sequence $\{u^{(0)}, u^{(1)}, \cdots, u^{(n)}, \cdots\}$ would be generated, where $u^{(0)} = f$. In each iteration, for a pixel $i$, it selects pixels from $\mathcal{N}_1(i) = \{i-1, i+1\}$ for averaging. The first thing is to find out the homogeneous set $\Omega_i$ given by

$$\Omega_i = \{j \in \mathcal{N}_1(i) : |u_j^{(n)} - u_i^{(n)}| \le T\},$$

where $T$ is a predefined threshold parameter. Then, we use pixels of $\Omega_i$ to compute $u_i^{(n+1)}$. The set $\Omega_i$ can be divided into the following four cases:

- **The first case**: $\Omega_i = \{i-1, i+1\}$;
- **The second case**: $\Omega_i = \{i-1\}$;
- **The third case**: $\Omega_i = \{i+1\}$;
- **The fourth case**: $\Omega_i = \emptyset$.

Accordingly, there are four different weighted average schemes proposed in [44]. For convenience of description, these weighted average schemes can be unified the following formula:

$$u_i^{(n+1)} = \sum_{j \in \Omega_i} p u_j^{(n)} + (1 - (\sharp\Omega_i)p) u_i^{(n)}, \tag{1}$$

where $0 < p \le \frac{1}{2}$ is the weight parameter, and $\sharp\Omega_i$ denotes the number of elements of $\Omega_i$. Since the weights of $u_{i-1}^{(n)}$, $u_i^{(n)}$ and $u_{i+1}^{(n)}$ depend on the set $\Omega_i$, each case has different weights in (1). As pointed out in [44], the scheme (1) for the first case can be considered as a discrete approximation of the linear diffusion; the (1) for the second (or third) case corresponds to a discrete approximation of the linear diffusion with the Neumann boundary condition [41] at the right (or left) edge pixel; the (1) for the fourth case is a discrete approximation of the linear diffusion with two Neumann boundary conditions at both the right and left edge pixels. By (1), the GSAM prevents the diffusion between different homogeneous regions, and thus keep edges very well.

A drawback of GSAM [44] is that it only selects pixels from 1-neighborhood $\mathcal{N}_1(i)$ for updating the current value. Thus, it usually needs too many iterations to obtain satisfactory results in practical application. On the other hand, there exist sparse noisy pixels appearing in the output by GSAM, when the noise level is high.

2.2. **A selective averaging with multiple neighbors method for 1D signal.** To overcome the problems of GSAM [44] as discussed in Sect. 2.1, we propose a selective averaging with multiple neighbors method by choosing more homogeneous neighbors from a large domain. It depends on the property of the local geometry of signals. Similarly, a signal sequence $\{u^{(0)}, u^{(1)}, \cdots, u^{(n)}, \cdots\}$ will be generated, where $u^{(0)}$ is the observed $f$ and $u^{(n)}$ denotes the signal after the $n$-th iteration.

2.2.1. *The algorithm.* To update $u^{(n)}$, we start from its first pixel to the last one; see the computation order in Fig. 1. For a pixel $i$, let $\mathcal{N}_k^l(i)$ represent the set of its left $k$-neighbors ($k > 1$), i.e.,

$$\mathcal{N}_k^l(i) = \{j : i - k \le j < i\},$$

and $\mathcal{N}_k^r(i)$ represent the set of its right $k$-neighbors, i.e.,

$$\mathcal{N}_k^r(i) = \{j : i < j \le i + k\}.$$

The set $\mathcal{N}_k(i) = \mathcal{N}_k^l(i) \cup \mathcal{N}_k^r(i)$ denotes the whole search domain. Similar to GSAM, we select the homogeneous neighbors from $\mathcal{N}_k(i)$ to compute $u_i^{(n+1)}$. Denote $\Omega_k^l(i)$ and $\Omega_k^r(i)$ as the left and right homogeneous sets:

$$\Omega_k^l(i) = \begin{cases} \mathcal{N}_k^l(i) & \text{if } |u_i^{(n)} - u_j^{(n)}| \le T,\ \forall j \in \mathcal{N}_k^l(i), \\ \{j : i > j \ge i - j^l(i)\} & \text{otherwise;} \end{cases}$$

$$\Omega_k^r(i) = \begin{cases} \mathcal{N}_k^r(i) & \text{if } |u_i^{(n)} - u_j^{(n)}| \le T,\ \forall j \in \mathcal{N}_k^r(i), \\ \{j : i < j \le i + j^r(i)\} & \text{otherwise;} \end{cases}$$

where

$$\{j^l(i) : j^l(i) < k,\ |u_i^{(n)} - u_{i-j}^{(n)}| \le T,\ j = 1, 2, \cdots, j^l(i),\ |u_i^{(n)} - u_{i-j^l(i)-1}^{(n)}| > T\},$$

$$\{j^r(i) : j^r(i) < k,\ |u_i^{(n)} - u_{i+j}^{(n)}| \le T,\ j = 1, 2, \cdots, j^r(i),\ |u_i^{(n)} - u_{i+j^r(i)+1}^{(n)}| > T\}.$$

Note that, $j^l(i)$ helps to show which pixels of $\mathcal{N}_k^l(i)$ belong to the left homogeneous set $\Omega_k^l(i)$. Similarly, $j^r(i)$ helps to show which pixels of $\mathcal{N}_k^r(i)$ belong to the right homogeneous set $\Omega_k^r(i)$. We denote the index $i - K_i^l$ as the smallest element in $\Omega_k^l(i)$ and $i + K_i^r$ as the largest element in $\Omega_k^r(i)$. If $\Omega_k^l(i) = \mathcal{N}_k^l(i)$, then $K_i^l = k$; if $\Omega_k^l(i) = \{j : i > j \ge i - j^l(i)\}$, then $K_i^l = j^l(i)$ with $j^l(i) < k$. Similarly, we can obtain the value of $K_i^r$. Note, if $K_i^l = 0$ (or $K_i^r = 0$), then we set $\Omega_k^l(i) = \varnothing$ (or $\Omega_k^r(i) = \varnothing$). Let $\Omega_k(i) = \Omega_k^l(i) \cup \Omega_k^r(i)$ and $K_i = K_i^l + K_i^r$. Obviously, we have $\sharp\Omega_k(i) = K_i$.

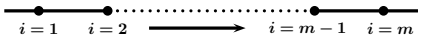


FIGURE 1. An illustration of the updating order from $i = 1$ to $i = m$.

For convenience of description, we introduce the following notations to represent different weighted average schemes:

$$\begin{aligned} v_1 &= \textstyle\sum_{j\in\Omega_k(i)} p u_j^{(n)} + (1 - K_i p) u_i^{(n)}, \\ v_2 &= \textstyle\sum_{j\in\Omega_k(i)} p u_j^{(n)} + p u_{i-K_i^l}^{(n)} + (1 - (K_i + 1)p) u_i^{(n)}, \\ v_3 &= \textstyle\sum_{j\in\Omega_k(i)} p u_j^{(n)} + p u_{i+K_i^r}^{(n)} + (1 - (K_i + 1)p) u_i^{(n)}, \\ v_4 &= \textstyle\sum_{j\in\Omega_k(i)} p u_j^{(n)} + p(u_{i-K_i^l}^{(n)} + u_{i+K_i^r}^{(n)}) + (1 - (K_i + 2)p) u_i^{(n)}, \end{aligned}$$

where $0 < p \le \frac{1}{2k}$ denotes the weight parameter in $v_1$, $v_2$, $v_3$ and $v_4$. The sum of weights is 1 for each $v_j$ ($j = 1, \cdots, 4$). In $v_2$, we add one more $u_{i-K_i^l}^{(n)}$, because we assume that there exists a Neumann boundary condition [41] at the edge pixel $i - K_i^l$, namely $\frac{\partial u^{(n)}}{\partial \hat{n}}\big|_{i-K_i^l} = 0$, where $\hat{n}$ is the outward normal at $i - K_i^l$. Similar to $v_2$, one more $u_{i+K_i^r}^{(n)}$ is added in $v_3$. In $v_4$, we assume that there exist two Neumann

boundary conditions at edge pixels $i-K_i^l$ and $i+K_i^r$, and thus one more $u^{(n)}_{i-K_i^l}$ and $u^{(n)}_{i+K_i^r}$ are added.

In the following, we present the method to select homogeneous neighbors of $i$-th pixel based on that of $i-1$. Since the indexes $K_{i-1}^l$ and $K_{i-1}^r$ at the pixel $i-1$ are known, we can find out $K_i^l$ and $K_i^r$ based on them. Note that, all $K_{i-1}^l$, $K_{i-1}^r$, $K_i^l$ and $K_i^r$ are the indexes in step $n$. There are five cases to analyze:

$$\begin{cases} \textbf{Case 1:} & K_{i-1}^r = 0; \\ \textbf{Case 2:} & K_{i-1}^l \geq k-1,\ K_{i-1}^r = k; \\ \textbf{Case 3:} & K_{i-1}^l < k-1,\ K_{i-1}^r = k; \\ \textbf{Case 4:} & K_{i-1}^l \geq k-1,\ 0 < K_{i-1}^r < k; \\ \textbf{Case 5:} & K_{i-1}^l < k-1,\ 0 < K_{i-1}^r < k. \end{cases} \tag{2}$$

According to the above different cases, we can obtain different $K_i^l$ and $K_i^r$ and the corresponding weighted average scheme.

The updated values $u_i^{(n+1)}$ in the above five cases are given as follows, where the detailed calculation process can be found in Appendix 6.

- **Case 1**:

$$u_i^{(n+1)} = \begin{cases} v_2, & \text{if } K_i^l = 0,\ K_i^r = k, \\ v_4, & \text{if } K_i^l = 0,\ K_i^r < k. \end{cases} \tag{7a}$$

- **Case 2**:

$$u_i^{(n+1)} = \begin{cases} v_1, & \text{if } K_i^l = k,\ K_i^r = k, \\ v_3, & \text{if } K_i^l = k,\ K_i^r = k-1. \end{cases} \tag{8a}$$

- **Case 3**:

$$u_i^{(n+1)} = \begin{cases} v_2, & \text{if } K_i^l = K_{i-1}^l + 1,\ K_i^r = k, \\ v_4, & \text{if } K_i^l = K_{i-1}^l + 1,\ K_i^r = k-1. \end{cases} \tag{9a}$$

- **Case 4**:

$$u_i^{(n+1)} = v_3, \quad \text{if } K_i^l = k,\ K_i^r = K_{i-1}^r - 1. \tag{10}$$

- **Case 5**:

$$u_i^{(n+1)} = v_4, \quad \text{if } K_i^l = K_{i-1}^l + 1,\ K_i^r = K_{i-1}^r - 1. \tag{11}$$

Using the above method for each pixel of $u^{(n)}$, we obtain a new signal $u^{(n+1)}$. Note that, we use the same $p$ value for each pixel. The corresponding iterative procedure is summarized in Algorithm 1.

**Algorithm 1** 1D Selective Averaging with Multiple Neighbors Method (SAMNM)

1: **Input**: Signal $f$, parameters $T$, $k$, $p$ and $\epsilon$.
2: **Initialization**: Set $u^{(0)} = f$.
3: **For** $n = 0, 1, 2, ...,$ compute $u^{(n+1)}$: for $i$, from 1 to $m$,
compute $u_i^{(n+1)}$ from (7a) or (7b) or (8a) or (8b) or (9a) or (9b) or (10) or (11).
4: **Until** $\frac{\|u^{(n+1)}-u^{(n)}\|}{\|u^{(n+1)}\|} < \epsilon$.
5: **Output**: $u^{(n+1)}$.

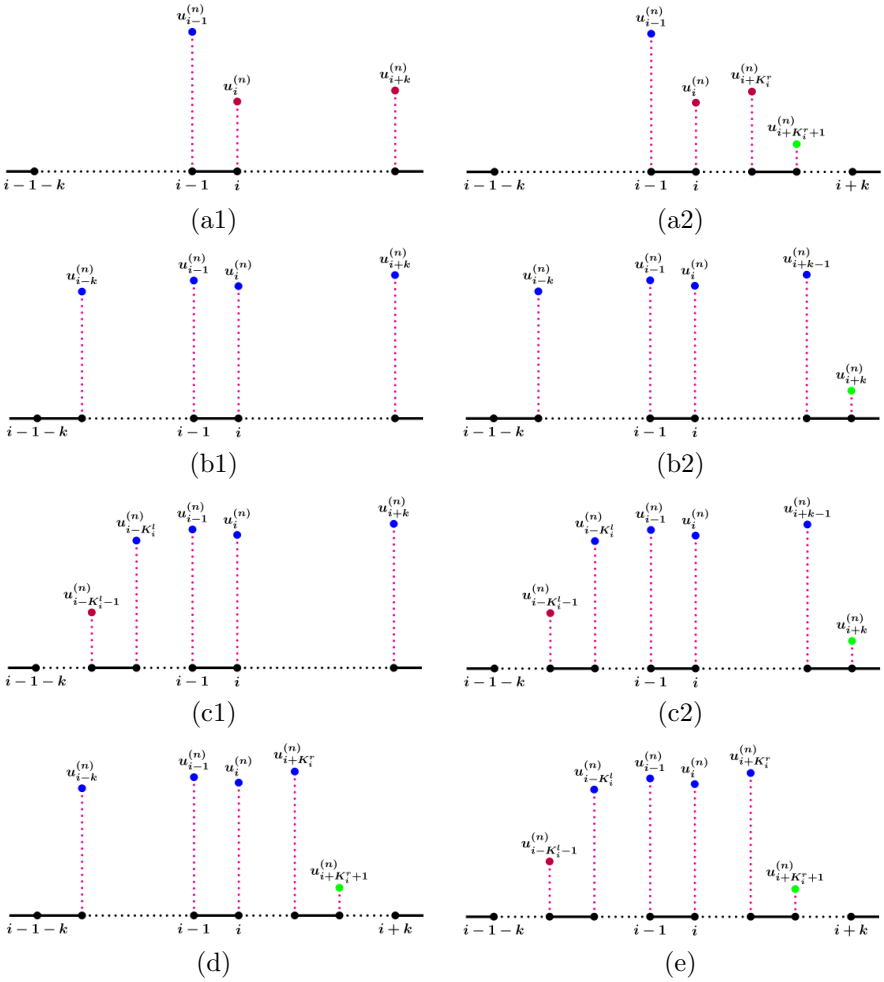


FIGURE 2. Different types to compute $u_i^{(n+1)}$ based on five cases of $K_{i-1}^l$ and $K_{i-1}^r$ from (2) to (6). (a1-a2), (b1-b2), (c1-c2), (d) and (e) correspond to Case 1, Case 2, Case 3, Case 4 and Case 5, respectively.

2.2.2. *Theoretical analysis.* We first define a jump vector $J^{(n)}$ corresponding to $u^{(n)}$. For any $i \in \{2, \cdots, m\}$, if $|u_{i-1}^{(n)} - u_i^{(n)}| > T$, we say that there is a jump (an edge) between $u_{i-1}^{(n)}$ and $u_i^{(n)}$, and define $J_i^{(n)} = 1$. If not, let it be 0. Obviously, $J^{(n)}$ includes $m-1$ elements. Meanwhile, $u^{(n)}$ is separated by the jumps into several segments. We first give a lemma to show that the jump vectors $\{J^{(n)}\}_{n=0}^{\infty}$ converges in a finite number of iterations. Then, we present the convergence result of the sequence $\{u^{(n)}\}_{n=0}^{\infty}$.

**Lemma 2.1.** *In Algorithm 1, no new jumps will be generated after each iteration, i.e., $J_i^{(n+1)} \leq J_i^{(n)}$, for $\forall i \in \{2, \cdots, m\}$. Furthermore, there exists an integer $N$ such that for $\forall n \geq N$, we have $J^{(n)} = J^{(N)}$. That is, $\{J^{(n)}\}_{n=1}^{\infty}$ converges in a finite number of iterations.*

*Proof.* We first prove $J_i^{(n+1)} \le J_i^{(n)}$, for $\forall i \in \{2, \cdots, m\}$. Assume that $|u_{i-1}^{(n)} - u_i^{(n)}| \le T$. We'd like to show $|u_{i-1}^{(n+1)} - u_i^{(n+1)}| \le T$. Note that, $|u_{i-1}^{(n+1)} - u_i^{(n+1)}| > T$ in Case 1, we do not consider this case. We just compute $u_{i-1}^{(n+1)}$ and $u_i^{(n+1)}$ based on four cases from Case 2 to Case 5 ((3), (4), (5), (6)) in Sect. 2.2.1.

To simplify describing the updated value $u_{i-1}^{(n+1)}$, we introduce some notations:

$$\begin{aligned}
\hat{v}_1 &= \textstyle\sum_{j\in\Omega_k(i-1)} p u_j^{(n)} + (1 - K_{i-1}p)u_{i-1}^{(n)},\\
\hat{v}_2 &= \textstyle\sum_{j\in\Omega_k(i-1)} p u_j^{(n)} + p u_{i-1-K_{i-1}^l}^{(n)} + (1 - (K_{i-1}+1)p)u_{i-1}^{(n)},\\
\hat{v}_3 &= \textstyle\sum_{j\in\Omega_k(i-1)} p u_j^{(n)} + p u_{i-1+K_i^r}^{(n)} + (1 - (K_{i-1}+1)p)u_{i-1}^{(n)},\\
\hat{v}_4 &= \textstyle\sum_{j\in\Omega_k(i-1)} p u_j^{(n)} + p(u_{i-1-K_{i-1}^l}^{(n)} + u_{i-1+K_{i-1}^r}^{(n)}) + (1 - (K_{i-1}+2)p)u_{i-1}^{(n)},
\end{aligned}$$

where $p$ is the weight parameter.

If $K_{i-1}^l$ and $K_{i-1}^r$ satisfy Case 2 in (3), then $u_{i-1}^{(n+1)}$ is computed by

$$u_{i-1}^{(n+1)} = \begin{cases} \hat{v}_2, & \text{if } K_{i-1}^l = k-1,\ K_{i-1}^r = k, \quad \text{(12a)}\\ \hat{v}_1, & \text{if } K_{i-1}^l = k,\ K_{i-1}^r = k. \quad \text{(12b)} \end{cases}$$

Note, when $K_{i-1}^l = k-1$, there is an edge between $(i-1)-(k-1)$ and $(i-1)-k$. Thus, we adopt the Neumann boundary conditions [41] at $(i-1)-K_{i-1}^l$.

If $K_{i-1}^l$ and $K_{i-1}^r$ satisfy Case 3 in (4), then we obtain $u_{i-1}^{(n+1)}$ by

$$u_{i-1}^{(n+1)} = \hat{v}_2, \quad \text{for } K_{i-1}^l < k-1,\ K_{i-1}^r = k. \tag{13}$$

If $K_{i-1}^l$ and $K_{i-1}^r$ satisfy Case 4 in (5), then $u_{i-1}^{(n+1)}$ is similarly computed by

$$u_{i-1}^{(n+1)} = \begin{cases} \hat{v}_4, & \text{if } K_{i-1}^l = k-1,\ 0 < K_{i-1}^r < k, \quad \text{(14a)}\\ \hat{v}_3, & \text{if } K_{i-1}^l = k,\ 0 < K_{i-1}^r < k. \quad \text{(14b)} \end{cases}$$

If $K_{i-1}^l$ and $K_{i-1}^r$ satisfy Case 5 in (6), then we obtain $u_{i-1}^{(n+1)}$ by

$$u_{i-1}^{(n+1)} = \hat{v}_4, \quad \text{for } K_{i-1}^l < k-1,\ 0 < K_{i-1}^r < k. \tag{15}$$

To compute $|u_{i-1}^{(n+1)} - u_i^{(n+1)}|$, there are nine types to analyze:
Type 1: If $K_{i-1}^l = k-1,\ K_{i-1}^r = k,\ K_i^l = k,\ K_i^r = k$ in Case 2, then we have from (8a) and (12a)

$$\begin{aligned}
&|u_{i-1}^{(n+1)} - u_i^{(n+1)}|\\
&= |(p u_{i-1-K_{i-1}^l}^{(n)} + (1 - (K_{i-1}+1)p)u_{i-1}^{(n)} + p u_i^{(n)})\\
&- (p u_{i-1}^{(n)} + (1 - K_i p)u_i^{(n)} + p u_{i+K_i^r}^{(n)})|\\
&= |p(u_{i-1-K_{i-1}^l}^{(n)} - u_{i-1}^{(n)}) + p(u_i^{(n)} - u_{i+K_i^r}^{(n)}) + (1 - K_i p)(u_{i-1}^{(n)} - u_i^{(n)})|\\
&\le pT + pT + (1 - K_i p)T < T.
\end{aligned}$$

Type 2: If $K_{i-1}^l = k-1,\ K_{i-1}^r = k,\ K_i^l = k,\ K_i^r = k-1$ in Case 2, then we have from (8b) and (12a)

$$\begin{aligned}
&|u_{i-1}^{(n+1)} - u_i^{(n+1)}| \\
&= |(pu_{i-1-K_{i-1}^l}^{(n)} + (1-(K_{i-1}+1)p)u_{i-1}^{(n)} + pu_i^{(n)}) \\
&- (pu_{i-1}^{(n)} + (1-(K_i+1)p)u_i^{(n)} + pu_{i+K_i^r}^{(n)})| \\
&= |p(u_{i-1-K_{i-1}^l}^{(n)} - u_{i-1}^{(n)}) + p(u_i^{(n)} - u_{i+K_i^r}^{(n)}) + (1-(K_i+1)p)(u_{i-1}^{(n)} - u_i^{(n)})| \\
&\le pT + pT + (1-(K_i+1)p)T < T.
\end{aligned}$$

Type 3: If $K_{i-1}^l = k,\ K_{i-1}^r = k,\ K_i^l = k,\ K_i^r = k$ in Case 2, then we have from (8a) and (12b)

$$\begin{aligned}
&|u_{i-1}^{(n+1)} - u_i^{(n+1)}| \\
&= |(pu_{i-1-K_{i-1}^l}^{(n)} + (1-K_{i-1}p)u_{i-1}^{(n)} + pu_i^{(n)}) \\
&- (pu_{i-1}^{(n)} + (1-K_ip)u_i^{(n)} + pu_{i+K_i^r}^{(n)})| \\
&= |p(u_{i-1-K_{i-1}^l}^{(n)} - u_{i-1}^{(n)}) + p(u_i^{(n)} - u_{i+K_i^r}^{(n)}) + (1-K_ip)(u_{i-1}^{(n)} - u_i^{(n)})| \\
&\le pT + pT + (1-K_ip)T < T.
\end{aligned}$$

Type 4: If $K_{i-1}^l = k,\ K_{i-1}^r = k,\ K_i^l = k,\ K_i^r = k-1$ in Case 2, then we have from (8b) and (12b)

$$\begin{aligned}
&|u_{i-1}^{(n+1)} - u_i^{(n+1)}| \\
&= |(pu_{i-1-K_{i-1}^l}^{(n)} + (1-K_{i-1}p)u_{i-1}^{(n)} + pu_i^{(n)}) \\
&- (pu_{i-1}^{(n)} + (1-(K_i+1)p)u_i^{(n)} + pu_{i+K_i^r}^{(n)})| \\
&= |p(u_{i-1-K_{i-1}^l}^{(n)} - u_{i-1}^{(n)}) + p(u_i^{(n)} - u_{i+K_i^r}^{(n)}) + (1-(K_i+1)p)(u_{i-1}^{(n)} - u_i^{(n)})| \\
&\le pT + pT + (1-(K_i+1)p)T < T.
\end{aligned}$$

Type 5: If $K_{i-1}^l < k-1,\ K_{i-1}^r = k,\ K_i^l = K_{i-1}^l + 1,\ K_i^r = k$ in Case 3, then we have from (9a) and (13)

$$\begin{aligned}
|u_{i-1}^{(n+1)} - u_i^{(n+1)}| &= |((1-(K_{i-1}+1)p)u_{i-1}^{(n)} + pu_i^{(n)}) \\
&- (pu_{i-1}^{(n)} + (1-(K_i+1)p)u_i^{(n)} + pu_{i+K_i^r}^{(n)})| \\
&= |p(u_i^{(n)} - u_{i+K_i^r}^{(n)}) + (1-(K_i+1)p)(u_{i-1}^{(n)} - u_i^{(n)})| \\
&\le pT + (1-(K_i+1)p)T < T.
\end{aligned}$$

Type 6: If $K_{i-1}^l < k-1,\ K_{i-1}^r = k,\ K_i^l = K_{i-1}^l + 1,\ K_i^r = k-1$ in Case 3, then we have from (9b) and (13)

$$\begin{aligned}
|u_{i-1}^{(n+1)} - u_i^{(n+1)}| &= |((1-(K_{i-1}+1)p)u_{i-1}^{(n)} + pu_i^{(n)}) \\
&- (pu_{i-1}^{(n)} + (1-(K_i+2)p)u_i^{(n)} + pu_{i+K_i^r}^{(n)})| \\
&= |p(u_i^{(n)} - u_{i+K_i^l}^{(n)}) + (1-(K_i+2)p)(u_{i-1}^{(n)} - u_i^{(n)})| \\
&\le pT + (1-(K_i+2)p)T < T.
\end{aligned}$$

Type 7: If $K_{i-1}^l = k-1,\ 0 < K_{i-1}^r < k,\ K_i^l = k,\ K_i^r = K_{i-1}^r - 1$ in Case 4, then we have from (10) and (14a)

$$\begin{aligned}|u_{i-1}^{(n+1)} - u_i^{(n+1)}| &= |(pu_{i-1-K_{i-1}^l}^{(n)} + (1-(K_{i-1}+2)p)u_{i-1}^{(n)} + pu_i^{(n)}) \\ &\quad - (pu_{i-1}^{(n)} + (1-(K_i+1)p)u_i^{(n)})| \\ &= |p(u_{i-1-K_{i-1}^l}^{(n)} - u_{i-1}^{(n)}) + (1-(K_i+2)p)(u_{i-1}^{(n)} - u_i^{(n)})| \\ &\le pT + (1-(K_i+2)p)T < T.\end{aligned}$$

Type 8: If $K_{i-1}^l = k,\ 0 < K_{i-1}^r < k,\ K_i^l = k,\ K_i^r = K_{i-1}^r - 1$ in Case 4, then we have from (10) and (14b)

$$\begin{aligned}|u_{i-1}^{(n+1)} - u_i^{(n+1)}| &= |(pu_{i-1-K_{i-1}^l}^{(n)} + (1-(K_{i-1}+1)p)u_{i-1}^{(n)} + pu_i^{(n)}) \\ &\quad - (pu_{i-1}^{(n)} + (1-(K_i+1)p)u_i^{(n)})| \\ &= |p(u_{i-1-K_{i-1}^l}^{(n)} - u_{i-1}^{(n)}) + (1-(K_i+2)p)(u_{i-1}^{(n)} - u_i^{(n)})| \\ &\le pT + (1-(K_i+2)p)T < T.\end{aligned}$$

Type 9: If $K_{i-1}^l < k-1,\ 0 < K_{i-1}^r < k,\ K_i^l = K_{i-1}^l + 1,\ K_i^r = K_{i-1}^r - 1$ in Case 5, then we have from (11) and (15)

$$\begin{aligned}|u_{i-1}^{(n+1)} - u_i^{(n+1)}| &= |((1-(K_{i-1}+2)p)u_{i-1}^{(n)} + pu_i^{(n)}) \\ &\quad - (pu_{i-1}^{(n)} + (1-(K_i+2)p)u_i^{(n)})| \\ &= |(1-(K_{i-1}+3)p)(u_{i-1}^{(n)} - u_i^{(n)})| \\ &\le (1-(K_{i-1}+3)p)T < T.\end{aligned}$$

From these above nine types, it means that there is no jump between $u_{i-1}^{(n+1)}$ and $u_i^{(n+1)}$. Thus, no new jumps is generated after one iteration by Algorithm 1.

For $\forall i \in \{2, \cdots, m\}$, since $\{J_i^{(n)}\}_{n=0}^{\infty}$ forms a nonincreasing sequence and each term can only take 0 or 1, $\{J_i^{(n)}\}_{n=0}^{\infty}$ converges in a finite number of iterations. Accordingly, $\{J^{(n)}\}_{n=0}^{\infty}$ converges in a finite number of iterations. That is, there exists an integer $N$ such that for $\forall n \ge N$, we have $J^{(n)} = J^{(N)}$. □

According to Lemma 2.1, we observe that the locations of 1 in $J^{(n)}$ are fixed from the $N$-th iteration. It implies that, for $\forall n \ge N$, $u^{(n)}$ is separated into several fixed segments (homogeneous regions), which are exactly the segments of $u^{(N)}$. Let $S^{(N)}$ denote the number of these fixed segments, and $u_s^{(n)} = \{u_{i_s}^{(n)}, u_{i_s+1}^{(n)}, \cdots, u_{i_s+t_s-1}^{(n)}\}$ denote pixels in the $s$-th segment, for $\forall s \in \{1, 2, \cdots, S^{(N)}\}$. As the GSAM in [44], the whole fixed weight matrix $W^{(N)}$ is also a block diagonal matrix consisting of $S^{(N)}$ blocks and can be written as

$$W^{(N)} = \begin{pmatrix} W_1 & 0 & \dots & 0 \\ 0 & W_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & W_{S^{(N)}} \end{pmatrix}. \tag{16}$$

For $\forall n \ge N$, we use the same weight parameter for pixels in each homogeneous region $u_s^{(n)}$. Theoretically, the total number of parameters is $S^{(N)}$. In this paper,

for convenience of description, we use the same parameter $p$ for all homogeneous regions. Thus, each block $W_s$ has the following form:

(17)
$$W_s = \begin{pmatrix} 1-kp & p & \dots & p & 0 & \dots & 0 & \dots & 0 & 0 \\ 2p & 1-(k+2)p & \dots & p & p & \dots & 0 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & 0 & 0 & \dots & p & \dots & 1-(k+2)p & 2p \\ 0 & 0 & \dots & 0 & 0 & \dots & 0 & \dots & p & 1-kp \end{pmatrix},$$

where $0 < p \leq \frac{1}{2k}$ and $W_s$ is a $t_s \times t_s$ non-negative matrix. For example, if $k = 2$ and $t_s = 5$, $W_s$ is given by

(18)
$$W_s = \begin{pmatrix} 1-2p & p & p & 0 & 0 \\ 2p & 1-4p & p & p & 0 \\ p & p & 1-4p & p & p \\ 0 & p & p & 1-4p & 2p \\ 0 & 0 & p & p & 1-2p \end{pmatrix}.$$

From (18), we see that the rows sum of $W_s$ is 1, and thus $W_s$ is called row-stochastic matrix. When $n \geq N$, the averaging procedure in each segment is simplified to

(19)
$$u_s^{(n)} = W_s^{n-N} u_s^{(N)}.$$

**Remark 1.** In the interior of each homogeneous region $u_s^{(n)}$, Algorithm 1 can be considered as a discrete approximation of the linear diffusion in a local domain, while a discrete approximation of the linear diffusion with the Neumann boundary condition [41] at edges.

To further analyze the property of each $W_s$, we introduce two basic concepts of graph theory:

**Definition 2.2.** A directed graph is an ordered pair $G = (V, A)$, where

- $V$ is a set whose elements are called vertices;
- $A$ is a set of ordered pairs of vertices, called directed edges.

**Definition 2.3.** A directed graph $G = (V, A)$ is called strongly connected if there is a path in each direction between each pair of vertices, i.e., for any two vertices $V_i$ and $V_j$, there exists a directed path connecting them.

We consider that $W_s$ is associated a finite directed graph $G_s$, where the number of vertices is $t_s$. Let $V_1$, $V_2$, $\cdots$, $V_{t_s}$ denote these distinct $t_s$ vertices. For each element $W_s(i,j) \neq 0$, we connect $V_i$ to $V_j$ by a directed edge $\overrightarrow{V_i V_j}$. It is possible to get to any vertex $V_i$ from any vertex $V_j$, for $1 \leq i, j \leq t_s$. Thus, the directed graph $G_s$ is strongly connected.

Now, we have the following theorem.

**Theorem 2.4.** *The sequence $\{u^{(n)}\}_{n=0}^{\infty}$ generated by Algorithm 1 converges to a signal consisting of several segments, where each segment is a constant vector.*

*Proof.* In order to show the convergence of $\{u^{(n)}\}_{n=0}^{\infty}$, we only need to prove the convergence of $\{u^{(n)}\}_{n=N}^{\infty}$. Note that, $N$ is a finite number given in Lemma 2.1. Without loss of generality, we consider the convergence of $\{u_s^{(n)}\}_{n=N}^{\infty}$ in each segment. A sufficient condition is that all the absolute eigenvalues of $W_s$ in (19) are

less than or equal to 1, and $-1$ is not an eigenvalue of $W_s$ ($(-1)^{n-N}$ is divergent). This is shown in the following.

We first prove that each absolute eigenvalue of $W_s$ is less than or equal to 1. Suppose that $\rho$ is an eigenvalue of $W_s$ and $\mathbf{y}$ is the corresponding eigenvector, i.e., $W_s\mathbf{y} = \rho\mathbf{y}$. Let $|y_j| = \|\mathbf{y}\|_\infty$ for some $j$. Since $\sum_{i=1}^{t_s} W_s(j,i) = 1$, we have $|\rho y_j| = |(W_s\mathbf{y})_j| \le (\sum_{i=1}^{t_s} W_s(j,i))|y_j| = |y_j|$. Accordingly, $|\rho| \le 1$.

We then show that 1 is the largest simple eigenvalue of $W_s$ and $-1$ is not an eigenvalue. It is obvious that 1 is an eigenvalue of $W_s$, because $W_s\mathbf{1} = \mathbf{1}$. Here, $\mathbf{1}$ denotes the all-ones vector. As mentioned above, $W_s$ is associated a strongly connected directed graph $G_s$. From Theorem 1.17 in [36], it indicates that $W_s$ is irreducible. Here, non-negative matrix $W_s$ mens that each element of $W_s$ is non-negative. According to Chap.8 in [25] (or Froebenius-Perron theorem) about the non-negative irreducible matrix, we have that 1 is the unique eigenvalue of $W_s$ with maximum modulus. Therefore, $-1$ is not an eigenvalue and 1 is the largest simple eigenvalue.

Based on these analyses, $\{{W_s}^{n-N}\}_{n=N}^{\infty}$ converges. This implies that $\{u_s^{(n)}\}_{n=N}^{\infty}$ converges to a constant vector, where each coordinate has the same value. Consequently, $\{u^{(n)}\}_{n=0}^{\infty}$ converges to a signal consisting of several segments, and each segment is a constant vector. □

In Fig. 3, the first and third rows show two examples to compare the proposed Algorithm 1 to GSAM [44], when the clean signal is corrupted by Gaussian noise with $\sigma = 0.12$ and $\sigma = 0.16$. The stopping condition is the same as that in Algorithm 1 for two algorithms, and $\epsilon = 10^{-6}$. Here, we adopt the same threshold parameter $T$ for them. As in [44], $p$ is fixed to 0.49. We tune $p$ and $k$ in Algorithm 1. From Fig. 3, two methods can obtain good results visually, when the noise level is low. However, in terms of SNR defined by (20), the proposed Algorithm 1 gains about 1.0 dB than the GSAM [44] quantificationally. When $\sigma = 0.16$, we found that there is a noisy pixel left in the final result by GSAM [44]. In contrast, Algorithm 1 achieves satisfactory performance, where the result is very close to the clean signal. Moreover, Algorithm 1 even exceeds the GSAM [44] by about 10.0 dB in terms of SNR. Table 1 further lists the comparison of the iterative number and time cost by GSAM [44] and our Algorithm 1. We found that Algorithm 1 always needs less iterations and is faster than GSAM [44]. Therefore, we can say that the proposed Algorithm 1 indeed gives better result with less time consumption than the previous GSAM [44].

TABLE 1. Iterative number and time comparison(in seconds) by GSAM [44] and Algorithm 1 in Fig. 3.

| Signal | $\sigma$ | GSAM [44] | | Proposed | |
|---|---|---|---|---|---|
| | | iter | t | iter | t |
| Fig. 3 | 0.12 | 6231 | 1.61 | 3695 | 1.07 |
| | 0.16 | 3322 | 0.98 | 2181 | 0.79 |

Since the total variation [37] (TV) and $L_0$ gradient minimization [47] ($L_0$) are very popular for piecewise constant and smooth signal and image denoising, we also compare them to our proposed algorithm. In the second and fourth rows of Fig. 3, we present the results by TV [37] and $L_0$ [47] for two noisy signals. One can see that

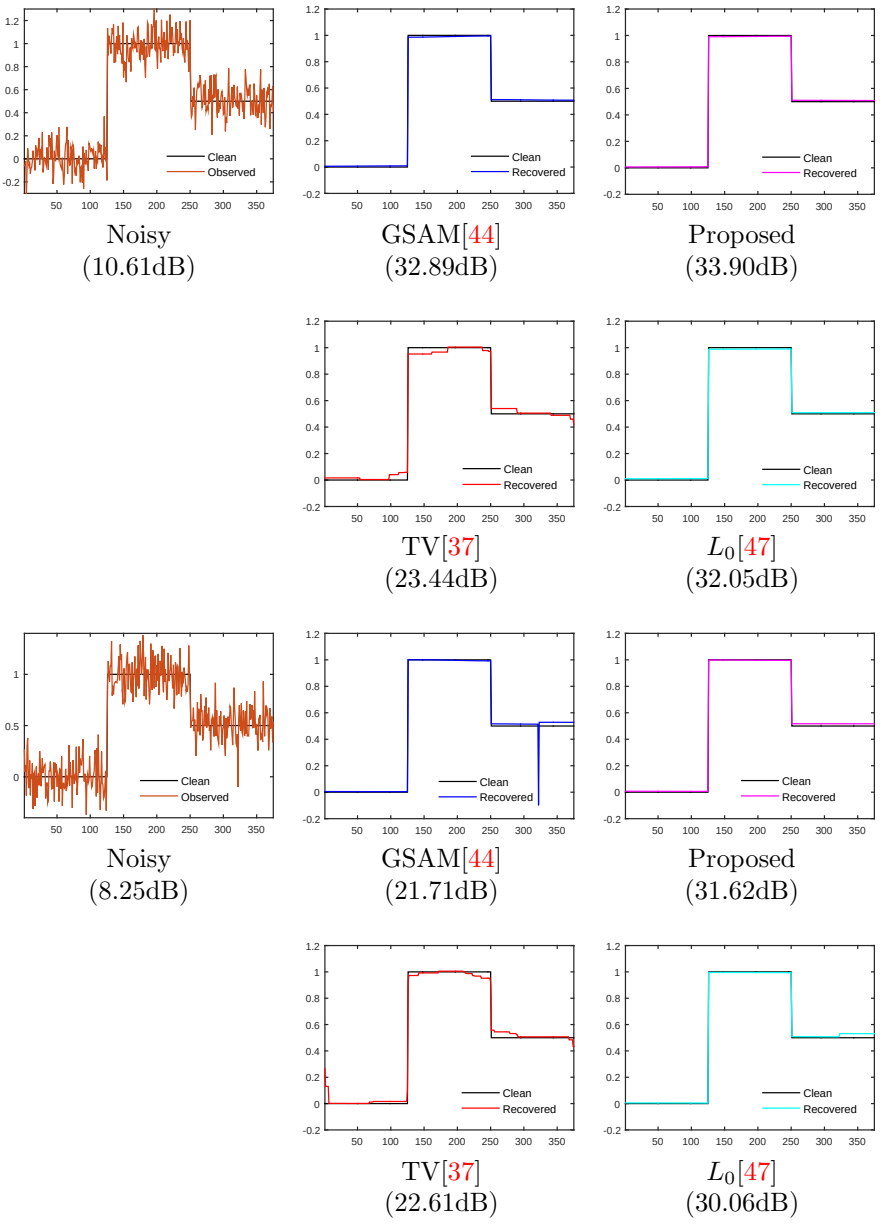


FIGURE 3. The first two rows show the results by GSAM [44] ($T = 0.44, p = 0.49$), Algorithm 1 ($T = 0.44, k = 2, p = 0.25$), TV[37] ($\alpha = 1.3, r_p = 10$) and $L_0$[47] ($\lambda = 0.03, \kappa = 1.01$), when $\sigma = 0.12$. The last two rows show the results by GSAM [44] ($T = 0.43, p = 0.49$), Algorithm 1 ($T = 0.43, k = 3, p = 0.11$), TV[37] ($\alpha = 1.4, r_p = 10$) and $L_0$[47] ($\lambda = 0.05, \kappa = 1.01$), when $\sigma = 0.16$. The corresponding SNR values are listed in brackets.

the two results by TV suffer from visible staircase effect. In contrast, $L_0$ obtains better result when $\sigma = 0.12$; but, it also causes staircase effect, when $\sigma = 0.16$. In terms of SNR, the proposed Algorithm 1 gains about 1.8 dB and 1.5 dB over $L_0$ from Figs. 3, respectively.

**Remark 2.** For any $s \in \{1, 2, \cdots, S^{(N)}\}$, the denoising iteration (19) can be considered corresponding a time-homogeneous Markov chain with a transition probability matrix $W_s$, because $W_s$ is a row-stochastic matrix in each fixed segment. The state space of each Markov chain is $\mathbf{E}_s = \{i_s, i_s + 1, \cdots, i_s + t_s - 1\}$. The denoised result $u_s^{(n+1)}$ at a pixel $i$ is the expected value of the Markov chain starting from the state $i$ to other states after $n$ iterations. Interested readers can refer to the probabilistic interpretation of GSAM [44] for details.

## 3. Extension to 2D case.

### 3.1. 2D gray image.

Without loss of generality, a piecewise constant gray image is represented as an $M_1 \times M_2$ matrix, where the index set is $\mathcal{B} = \{1, 2, \cdots, M_1\} \times \{1, 2, \cdots, M_2\}$. For a pixel $(i, j)$, we consider that the index $i$ corresponds to $x$-axis, and the index $j$ corresponds to $y$-axis. The reason is that an image is sampled from an underlying function of two real variables. Let $V$ denote the Euclidean space $\mathbb{R}^{M_1 \times M_2}$. Assume that an observed image $F$ is corrupted from the clean image $U$ by the zero-mean Gaussian noise with standard deviation $\sigma$.

In [44], we present an alternating general selective averaging method (AGSAM), which alternatively uses 1D GSAM in the $x$-axis and $y$-axis. We found that this method is quite suitable to 2D piecewise constant image denoising. Similarly, we alternatively apply Algorithm 1 (denoted by SAMNM) in the $x$-axis and $y$-axis. An image sequence $\{U^{(0)}, U^{(1)}, \cdots, U^{(n)}, \cdots\}$ will be generated, where $U^{(0)}$ is the observed image $F$ and $U^{(n)}$ denote the denoised image after the $n$-th iteration.

The following is the details of the alternating SAMNM (ASAMNM):

- **The first phase**: Compute $U^{(n+\frac{1}{2})}$ from $U^{(n)}$ by using Algorithm 1 in the $x$-axis. That is, for each row of $U^{(n)}$, call the SAMNM;
- **The second phase**: Compute $U^{(n+1)}$ from $U^{(n+\frac{1}{2})}$ by using Algorithm 1 in the $y$-axis. That is, for each column of $U^{(n+\frac{1}{2})}$, call the SAMNM.

There are three parameters: $T$, $k$ and $p$, as that in Algorithm 1. Similar to AGSAM in [44], we can only provide a numerical verification that $\{U^{(n)}\}_{n=0}^{\infty}$ converges to a piecewise constant image from experiments; see Sect. 4.

### 3.2. 2D color image.

An input color image denoted by $\mathbf{F}$ has three channels, i.e., $\mathbf{F} = (F_1, F_2, F_3)$. Each $F_m$ belongs to the Euclidean space $V$ for $m = 1, 2, 3$. The intensity at each pixel $(i, j)$ includes three values, namely, $\mathbf{F}_{i,j} = ((F_1)_{i,j}, (F_2)_{i,j}, (F_3)_{i,j})$. For any two pixels $\mathbf{F}_{i_1,j_1}$ and $\mathbf{F}_{i_2,j_2}$ of a color image, the absolute difference between them is defined as

$$|\mathbf{F}_{i_1,j_1} - \mathbf{F}_{i_2,j_2}| = \sqrt{\sum_{q=1}^{3}((F_q)_{i_1,j_1} - (F_q)_{i_2,j_2})^2}.$$

Once these values are computed, we can find the homogeneous pixels for each pixel. Accordingly, we choose the correct scheme for each pixel to update its value. Then, we iterate this process as ASAMNM in Sect. 3.1.

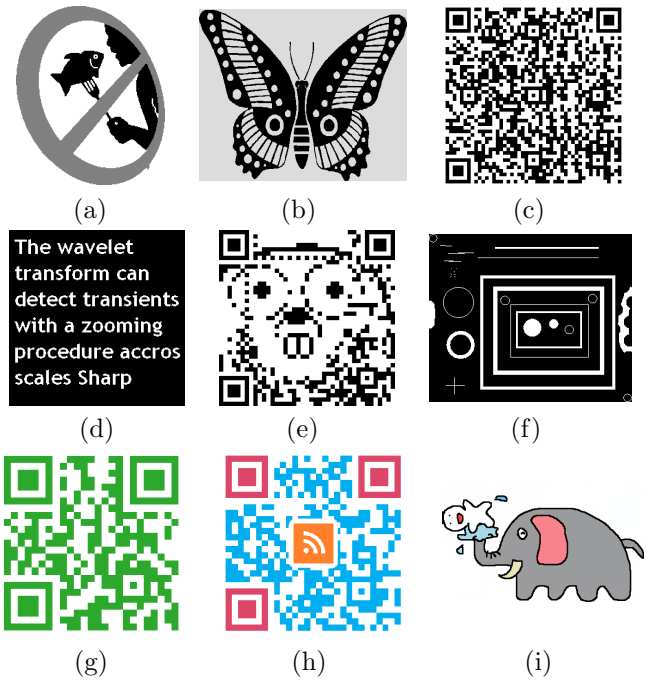


FIGURE 4. Test images. (a) Logo. (b) Cartoon1. (c) QRcode1. (d) Text. (e) QArtcode. (f) Blobs. (g) QRcode2. (h) QRcode3. (i) Cartoon2.

4. **Experiments results.** In this section, we mainly perform experiments and comparisons for eliminating additive Gaussian noise from piecewise constant gray and color images. Then, we try to apply our algorithm to speckle noise removal as the preliminary exploration. All the experiments are conducted using Windows 8 and MATLAB R2015a on a desktop with an Intel-i7 processor at 3.6GHz and 8GB RAM.

Except for visual quality evaluation, we also use the signal-to-noise ratio (SNR) defined by

$$\mathrm{SNR} = 10\log_{10}\frac{\|G-\tilde{G}\|_2}{\|G-U\|_2} \tag{20}$$

as the quantitative measurement to evaluate the performence of restoration, where $\tilde{G}$ is the mean intensity value of the original image $G$ and $U$ is the recovered image. Fig. 4 shows the clean test images.

4.1. **Gaussian noise removal.** For removing additive Gaussian noise, we compare our ASAMNM to TV [37, 1], $L_0$ [47], NLM-SAP [16] and AGSAM [44]. As we know, both TV and $L_0$ can effectively restore the cartoon part of images. The NLM-SAP adopts adaptive patch shapes instead of the usual square patches to reduce the noisy halos created around edges, and thus is suitable to recover piecewise constant images. The previous work of AGSAM also achieves good performance. We implemented the TV [37, 1] based on ADMM in [46], and used the code of AGSAM [44]. The source codes of $L_0$ [47] and NLM-SAP [16] were respectively downloaded

from their authors' homepages. Except $L_0$ and NLM-SAP using the termination conditions of their source codes, the remaining algorithms were terminated when $\frac{\|U^{(k+1)}-U^{(k)}\|}{\|U^{(k+1)}\|} < 10^{-5}$.

For each method, we tuned parameters carefully to achieve the best visual and quantitative results. In TV [37], we fixed a positive real number $r = 10$, and tuned a regularization parameter $\alpha$. In $L_0$ [47], we fixed a rate $\kappa = 1.01$ to allow more iterations for better results and tuned a smoothing weight $\lambda$. Considering computational performance and time cost of NLM-SAP [16], the size of search window was set $31 \times 31$ and the number of shapes was 24. Moreover, we tuned the bandwidth $h$ and the smoothing parameter $T$ in the aggregating strategy EWA. Since $p = 0.49$ in AGSAM [44], we tuned the threshold parameter $T$. Although there are three parameters in ASAMNM, we found that $k = 4$ is a good choice for all experiments, and we only tuned $T$ and $p$. In Table 2, we listed all the values of these methods for gray image denoising, while Table 5 gave all the values of these methods for color image denoising.

4.1.1. *Piecewise constant gray image denoising.* Fig. 5 shows the denoising results by different methods for the Logo corrupted by Gaussian noise with $\sigma = 15$. Visually, all results are good, where each algorithm is able to preserve sharp edges and eliminate most noise. One can see that $L_0$ [47] performs better for recovering flat-regions than TV [37], where TV suffers from the staircase effects. As pointed out in [10], both the results by TV [37] and $L_0$ [47] may include some noisy pixels near edges; see the zoom-in views. Although NLM-SAP is better than the classical NLM [4], those pixels near edges are not totally in accordance with the clean ones. This also can be found from the intensity values along a horizontal line in Fig. 6. In contrast, the AGSAM [44] and our ASAMNM can thoroughly eliminate noise in flat-regions and edges. Furthermore, it is seen from their zoom-in views that they produce piecewise constant results experimentally. From the quantitative SNR, the improvement by ASAMNM is 1.39 dB over AGSAM [44] (the second best method). The reason may be that, in finite iterations, more neighbors are used for averaging in ASAMNM generally would yield better result.

Fig. 6 shows results of a cartoon image with thin lines, and compares intensity values along part of a single row of each result. In this example, the main problem of TV [37] is that it fails to keep the intensity values in thin regions. One can see that there exists one noisy pixel at the edge of the recovered result by $L_0$ [47]. As discussed on Fig. 5, NLM-SAP [16] has the same problem when handling pixels at edges. The proposed ASAMNM and AGSAM [44] can overcome those problems of the first three algorithms, where the results are very close to the ground truth visually. However, our method still obtains the highest SNR, and outperforms the second best method (AGSAM [44]) by 1.79 dB.

To further demonstrate the validity of the proposed ASAMNM, another three experiments are shown in Figs. 7, 8 and 9. Except the example in Fig. 7, where $L_0$ [47] performs a little better than our method, the ASAMNM always obtains the best visual result and quantitative evaluation in the remaining two examples. The zoom-in views in the fourth column also gives an impression of this point.

Tables 3 and 4 list the SNR values, iterative number and runtime by different methods for other two noise levels in each set of experiments. In most cases, our method achieves the best SNR values. Since the number of iteration is 1 in the original code of NLM-SAP [16], we also adopts one iteration. However, NLM-SAP is slowest. The reason may be that its code is not optimized. Although our method

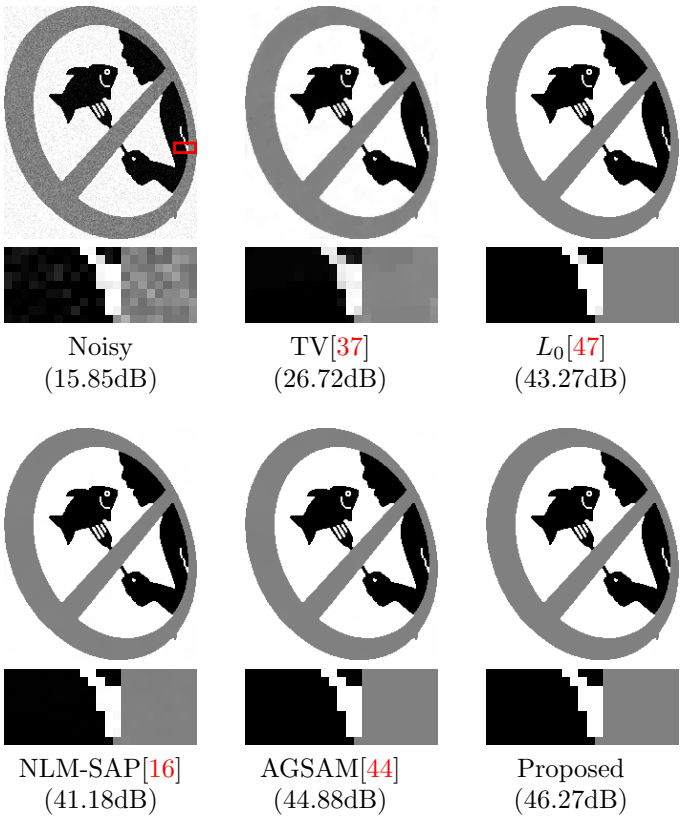


FIGURE 5. From top to down and left to right: the noisy Logo, the results by TV[37], $L_0$[47], NLM-SAP[16], AGSAM[44] and our method. The zoom-in views of the region indicited by the little red box in the top-left image are displayed for further comparison. The corresponding SNR values are listed in brackets.

needs a little more iterations than TV [37], it has the similar time cost with TV. Obviously, the iterations of ASAMNM are always less than AGSAM [44]. Since $\kappa = 1.01$ allows more iterations (over 1000 iterations) for piecewise constant image denoising in $L_0$ [47], its runtime is a little longer.

4.1.2. *Piecewise constant color image denoising.* In the following, we compare our method to TV [1], $L_0$ [47] and AGSAM [44] for color piecewise constant image denoising. Fig. 10 shows an example for noisy QRcode2 corrupted by Gaussian noise with $\sigma = 20$. Similar to the gray case, each method obtains good result visually. Yet, from the corresponding zoom-in views, TV [1] still fails to deal with those noisy pixels around edges. Although $L_0$ [47] has better noise reduction performance than TV, several noisy pixels at the corners are not eliminated. In this example, AGSAM [44] obtains similar visual result to our method, which can effectively preserve edges and remove noise very well. However, the proposed ASAMNM gives the noticeable improvement in terms of SNR over AGSAM [44].

Two other examples with higher noise level are presented in Fig. 11. Again, our method yields satisfactory results and performs best among all the compared

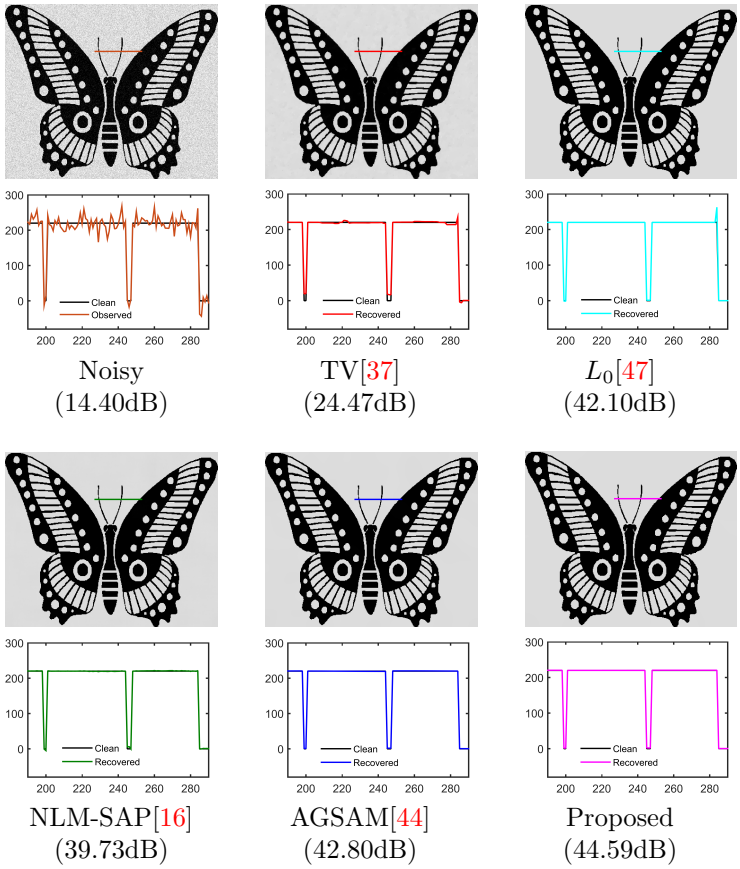


FIGURE 6. From top to down and left to right: the noisy Cartoon1, the results by TV[37], $L_0$[47], NLM-SAP[16], AGSAM[44] and our method. The zoom-in views of the region indicited by the little line in each image are displayed for further comparison. The corresponding SNR values are listed in brackets.

algorithms. To make a more comparison, we further add three noise levels for each set of experiments in Figs. 10 and 11. The SNR values, iterative number and runtime comparisons are summarized in Tables 6 and 7. The phenomenon in the two table is similar to that in Tables 3 and 4. Except the example of QRcode3 with $\sigma = 40$, the proposed method always achieves the highest SNR using shorter computation time.

4.2. **The preliminary exploration for speckle noise removal.** In ultrasound imaging, speckle noise is an important type of noise to affect accurate diagnosis. Lately, Wang et.al [43] proposed a novel model combining the advantage of total variation and high-order total variation. This approach overcomes staircase effect caused by TV regularization and is effective for suppressing speckle noise in ultrasound images. In this section, we just make a try to remove this kind of noise from piecewise constant images using the proposed ASAMNM, and the future work is to improve our method for real ultrasound image denoising. We compare our method

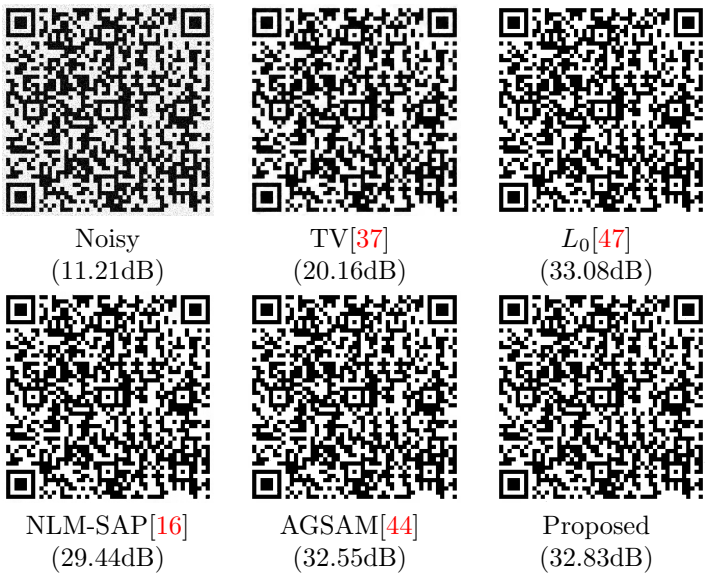


FIGURE 7. From top to down and left to right: the noisy QRcode1, the results by TV[37], $L_0$[47], NLM-SAP[16], AGSAM[44] and our method. The corresponding SNR values are listed in brackets.

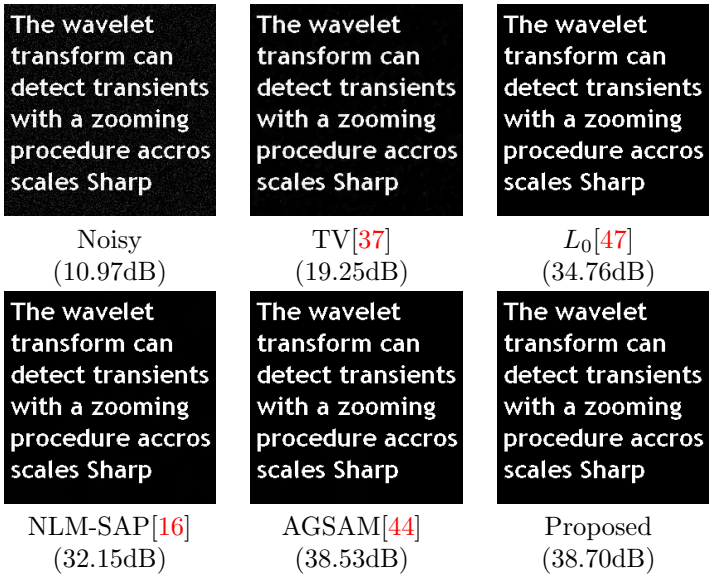


FIGURE 8. From top to down and left to right: the noisy Text, the results by TV[37], $L_0$[47], NLM-SAP[16], AGSAM[44] and our method. The corresponding SNR values are listed in brackets.

to Wang et.al [43], $L_0$ [47] and AGSAM [44]. As discussed in Wang et.al [43], the speckle noise can be modeled by

$$F = U + \sqrt{U}G,$$

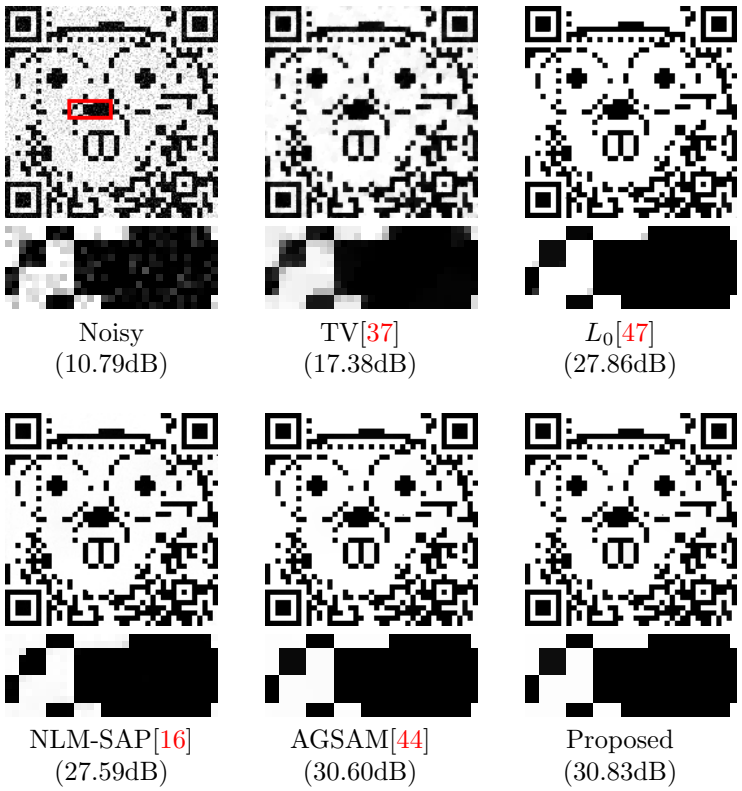


FIGURE 9. From top to down and left to right: the noisy QArtcode, the results by TV[37], $L_0$[47], NLM-SAP[16], AGSAM[44] and our method. The zoom-in views of the region indicited by the little red box in the top-left image are displayed for further comparison. The corresponding SNR values are listed in brackets.

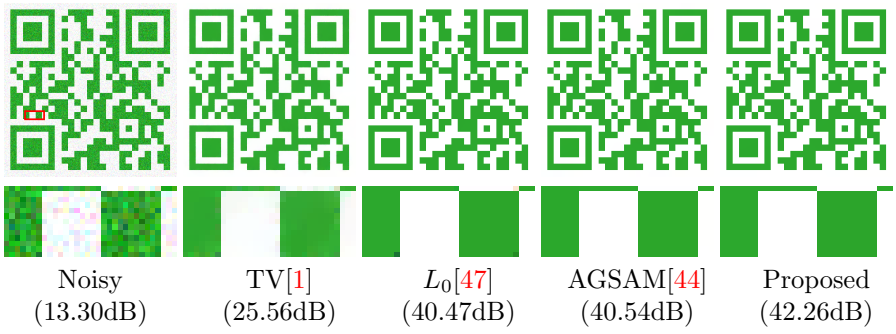


FIGURE 10. From left to right: the noisy QRcode2, the results by TV[1], $L_0$[47], AGSAM[44] and our method. The zoom-in views of the region indicited by the little red box in the first image are displayed for further comparison. The corresponding SNR values are listed in brackets.

TABLE 2. The parameter settings of different methods for gray images.

| Image | $\sigma$ | TV[37] $\alpha$ | $L_0$[47] $\lambda$ | NLM-SAP[16] $h/T$ | AGSAM[44] $T$ | Proposed $T/p$ |
|---|---|---|---|---|---|---|
| Logo | 10 | 30 | 0.01 | 1.5/0.6 | 5 | 5/0.13 |
| | 15 | 20 | 0.02 | 0.9/1.12 | 4 | 4/0.13 |
| | 20 | 14 | 0.03 | 0.7/2 | 2.4 | 3.3/0.05 |
| Cartoon1 | 15 | 20 | 0.02 | 1.6/1.35 | 5 | 5/0.13 |
| | 20 | 15 | 0.03 | 1.1/2.8 | 4.5 | 4.5/0.13 |
| | 25 | 12 | 0.04 | 0.8/5 | 4 | 4/0.06 |
| QRcode1 | 15 | 20 | 0.02 | 1.8/1.12 | 5 | 5/0.13 |
| | 25 | 12 | 0.04 | 1.1/3.12 | 4.5 | 4.5/0.13 |
| | 35 | 8 | 0.1 | 0.9/11.03 | 3.2 | 3.2/0.07 |
| Text | 15 | 24 | 0.02 | 1.7/1.8 | 4.5 | 4.5/0.13 |
| | 25 | 14 | 0.05 | 1.1/3.75 | 4.5 | 4.5/0.13 |
| | 35 | 10 | 0.06 | 1/9.8 | 3.2 | 3.5/0.13 |
| QArtcode | 15 | 23 | 0.02 | 1.7/1.8 | 5 | 5/0.13 |
| | 25 | 14 | 0.04 | 0.9/4.38 | 4.5 | 4.5/0.13 |
| | 35 | 10 | 0.07 | 0.6/9.8 | 3.4 | 3.4/0.07 |

TABLE 3. The SNR values by different methods for gray images.

| Image | $\sigma$ | TV[37] | $L_0$[47] | NLM-SAP[16] | AGSAM[44] | Proposed |
|---|---|---|---|---|---|---|
| Logo | 10 | 30.07 | 46.80 | 46.41 | 48.05 | **49.26** |
| | 20 | 24.41 | **39.15** | 34.96 | 29.05 | 34.77 |
| Cartoon1 | 15 | 26.86 | 44.60 | 43.22 | 44.95 | **46.87** |
| | 25 | 22.64 | 40.17 | 36.01 | 41.20 | **42.35** |
| QRcode1 | 15 | 27.20 | 41.98 | 42.00 | 43.78 | **44.97** |
| | 25 | 22.93 | 37.54 | 35.77 | 39.85 | **40.78** |
| Text | 15 | 23.55 | 39.19 | 39.75 | 42.57 | **43.06** |
| | 35 | 16.48 | **30.46** | 25.00 | 29.79 | 29.97 |
| QArtcode | 15 | 24.45 | 36.14 | **40.01** | 39.37 | 39.51 |
| | 25 | 20.16 | 31.70 | 35.08 | 35.20 | **35.25** |

where $F$ is the observed image, $U$ is the clean image and $G$ is the zero-mean Gaussian noise with standard deviation $\sigma$. Obviously, this type of noise is dependent on the intensity values of images. The parameter settings for $L_0$ [47], AGSAM [44] and our method are similar to that for additive Gaussian noise removal in Sect. 4.1. According to the code (provided by its authors) of Wang et.al [43], we tuned two parameters $\lambda$ and $\mu$. Table 8 gave all the values of these methods for the following experiments.

Fig. 12 presents a toy example when the "Blobs" image (built-in image from MATALB) is corrupted by speckle noise with $\sigma = 2$. The intensity values along part of a single column at the edge of each result are shown for further comparison. In this example, one can see that Wang et.al [43] only weakens the noise intensity and can not remove them thoroughly. The reason may be that TV-based regularization is not suitable to thin regions. Although the result by $L_0$ [47] looks better visually, there still exist sparse noisy pixels. We found that AGSAM [44] produces the

TABLE 4. The iterative number and time comparison(in seconds) by different methods for gray images.

| Image | $\sigma$ | TV[37] iter/t | $L_0$[47] iter/t | NLM-SAP[16] iter/t | AGSAM[44] iter/t | Proposed iter/t |
|---|---|---|---|---|---|---|
| Logo | 10 | 90/0.7 | 1551/5.5 | 1/141.4 | 139/0.5 | **66**/**0.2** |
| | 20 | **91**/0.6 | 1481/5.0 | 1/145.7 | 182/0.6 | 138/**0.4** |
| Cartoon1 | 15 | **96**/2.5 | 1481/19.6 | 1/698.3 | 170/3.5 | 102/**2.2** |
| | 25 | **98**/**2.6** | 1355/18.2 | 1/720.5 | 235/5.0 | 177/3.8 |
| QRcode1 | 15 | **83**/**2.2** | 1479/18.0 | 1/994.1 | 177/3.6 | 103/2.3 |
| | 25 | **84**/**2.1** | 1411/17.2 | 1/998.0 | 254/5.2 | 150/3.5 |
| Text | 15 | **112**/**0.7** | 1482/3.9 | 1/165.9 | 295/1.6 | 147/0.8 |
| | 35 | **109**/**0.6** | 1355/3.5 | 1/166.4 | 502/2.7 | 324/1.8 |
| QArtcode | 15 | **67**/0.3 | 1478/2.9 | 1/77.6 | 175/0.3 | 109/**0.2** |
| | 25 | **69**/0.3 | 1389/2.3 | 1/78.3 | 254/0.4 | 155/**0.2** |

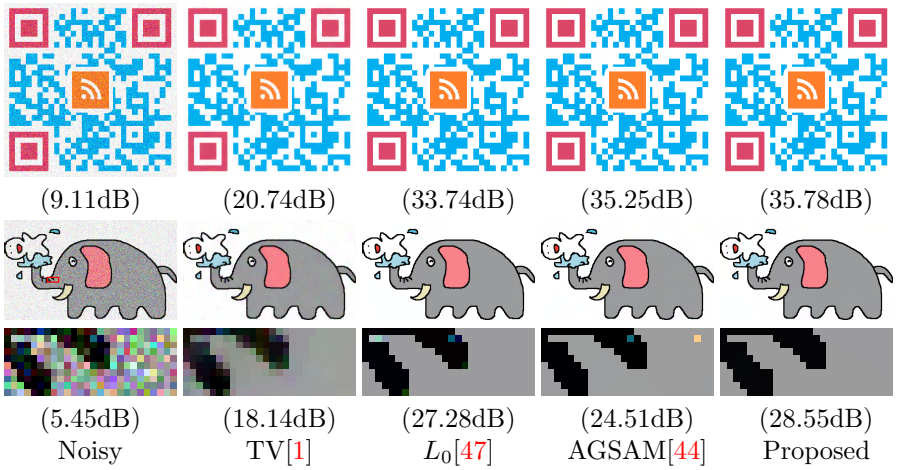


FIGURE 11. From up to down: Denoising results by different methods for two noisy QRcode3 and Cartoon2 images. From left to right: the noisy images, the results by TV[1], $L_0$[47], AGSAM[44] and our method. The zoom-in views of the region indicited by the little red box in the second row-left image are displayed for further comparison. The corresponding SNR values are listed in brackets.

comparable result with the proposed ASAMNM, where both them exceed the first two algorithms. Thus, we can say that the idea of selective averaging based on the Neumann boundary condition is quite effective for this "Blobs" image with many thin regions.

Two other examples are further shown in Fig. 13, when the Cartoon1 is corrupted by speckle noise with $\sigma = 2$ and 2.5 respectively. When $\sigma = 2$, we observe that our method performs best among all the compared methods, and exceeds the second best method (AGSAM [44]) by 1.1 dB in SNR. However, though the proposed ASAMNM is still better than Wang et.al [43] and AGSAM [44], it is worse than $L_0$ [47] when the level of noise is 2.5. Nevertheless, it is worth improving the ASAMNM

TABLE 5. The parameter settings of different methods for color images.

| Image | $\sigma$ | TV[1] $\alpha$ | $L_0$[47] $\lambda$ | AGSAM[44] $T$ | Proposed $T/p$ |
|---|---|---|---|---|---|
| QRcode2 | 10 | 16 | 0.01 | 5 | 5/0.13 |
| | 20 | 8 | 0.04 | 4 | 4/0.13 |
| | 30 | 5 | 0.08 | 3.5 | 3.5/0.13 |
| | 40 | 4 | 1.6 | 2.3 | 2.6/0.1 |
| QRcode3 | 10 | 17 | 0.02 | 5 | 5/0.13 |
| | 20 | 9 | 0.04 | 4 | 4/0.13 |
| | 30 | 6 | 0.12 | 2.5 | 2.6/0.11 |
| | 40 | 4 | 0.5 | 2 | 2/0.06 |
| Cartoon2 | 10 | 19 | 0.01 | 5 | 5/0.13 |
| | 20 | 10 | 0.05 | 4 | 4.2/0.13 |
| | 30 | 6 | 0.09 | 2.5 | 2.8/0.13 |
| | 40 | 5 | 1.16 | 1.9 | 2.3/0.1 |

TABLE 6. The SNR values by different methods for color images.

| Image | $\sigma$ | TV[1] | $L_0$[47] | AGSAM[44] | Proposed |
|---|---|---|---|---|---|
| QRcode2 | 10 | 31.29 | 46.49 | 46.02 | **47.62** |
| | 30 | 22.23 | 36.37 | 37.53 | **37.81** |
| | 40 | 20.01 | 29.14 | 33.39 | **33.85** |
| QRcode3 | 10 | 29.68 | 44.10 | 44.44 | **46.11** |
| | 20 | 23.97 | 38.08 | 38.91 | **40.77** |
| | 40 | 18.48 | **25.57** | 21.10 | 25.30 |
| Cartoon2 | 10 | 29.39 | 40.31 | 40.93 | **41.05** |
| | 20 | 23.65 | 38.36 | 38.46 | **39.53** |
| | 30 | 20.37 | 36.09 | 35.80 | **36.99** |

TABLE 7. The iterative number and time comparison(in seconds) by different methods for color images.

| Image | $\sigma$ | TV[1] iter/t | $L_0$[47] iter/t | AGSAM[44] iter/t | Proposed iter/t |
|---|---|---|---|---|---|
| QRcode2 | 10 | 76/3.0 | 1551/33.3 | 144/2.7 | **68**/**1.4** |
| | 30 | **82**/3.3 | 1342/29.6 | 253/3.4 | 172/**2.9** |
| | 40 | **75**/**3.0** | 1041/23.9 | 311/5.8 | 215/4.4 |
| QRcode3 | 10 | **66**/2.6 | 1411/31.3 | 139/2.7 | 68/**1.5** |
| | 20 | **63**/2.8 | 1340/32.3 | 183/3.5 | 118/**2.7** |
| | 40 | **72**/**2.9** | 1158/25.9 | 308/6.0 | 283/6.1 |
| Cartoon2 | 10 | **70**/3.4 | 1551/46.1 | 146/3.4 | 79/**1.9** |
| | 20 | **54**/3.2 | 1355/44.1 | 189/4.2 | 116/**2.8** |
| | 30 | **78**/3.8 | 1240/39.5 | 245/5.4 | 152/**3.7** |

for the real ultrasound image denoising, because it adopts the Neumann boundary condition at edges to prevent the diffusion between different regions.

TABLE 8. The parameter settings of different methods for gray images with speckle noise.

| Image | $\sigma$ | Wang et.al [43] $\lambda/\mu$ | $L_0$[47] $\lambda$ | AGSAM[44] $T$ | Proposed $T/p/k$ |
|---|---|---|---|---|---|
| Blobs | 2 | 1.2/11 | 0.04 | 60 | 60/0.04/12 |
| Cartoon1 | 2 | 0.8/2 | 0.05 | 58 | 58/0.13/4 |
| | 2.5 | 0.6/2 | 0.06 | 45 | 45/0.03/5 |

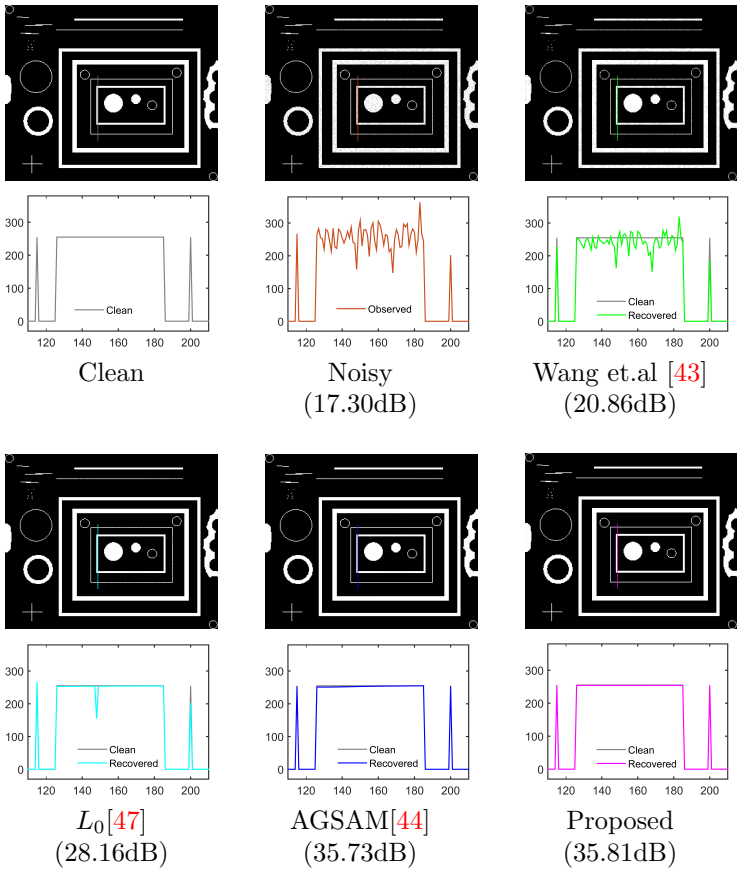


FIGURE 12. From top to down and left to right: the clean and noisy "Blobs", the results by Wang et.al [43], $L_0$ [47], AGSAM [44] and our method. The zoom-in views of the region indicited by the little line in the each image are displayed for further comparison. The corresponding SNR values are listed in brackets.

5. **Conclusion.** Piecewise constant signals and images are an important kind of data, which have wide applications in both commercial and automotive industry use. In this paper, we proposed a selective averaging with multiple neighbors method to eliminate noise from this type of data. To illustrate the essential idea, our algorithm was first presented in 1D case. Based on the property of the local geometry of signals, we select more homogeneous neighbors for averaging. Meanwhile, our method

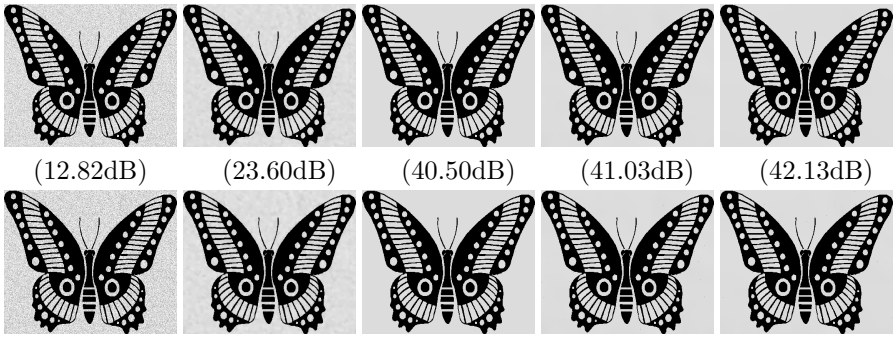


FIGURE 13. From up to down: Denoising results by different methods for Cartoon1 corrupted by two levels of noise. From left to right: the noisy images, the results by Wang et.al [43], $L_0$ [47], AGSAM [44] and our method. The corresponding SNR values are listed in brackets.

adopts the Neumann boundary condition at edges, and thus keeps edge very well. Compared to the GSAM in [44], the proposed algorithm can not only improve the efficiency of removing noise, but also greatly reduce sparse noisy pixels left in the final result. Furthermore, we proved that the iterative signal sequence converges to a signal consisting of several segments, where each segment is a constant vector. By alternatively applying the proposed algorithm in 1D case, the ASAMNM can be easily obtained for 2D image denoising.

Experimental results on both gray and color image denoising demonstrated that our algorithm is quite effective, and in most cases achieves superior performance from visual image quality and quantitative SNR evaluation. In particular, it always performs better than the AGSAM [44] with shorter time, under the same stopping condition. Although we mainly consider eliminating additive Gaussian noise or speckle noise from piecewise constant images, we believe that there is large room to improve ASAMNM for natural image and real ultrasound image denoising. To handle very high level of noise, it is also a future work.

6. **Appendix.** The detailed calculation process of (7a), (7b), (8a), (8b), (9a), (9b), (10) and (11) is given in the following.

- **Case 1**: Case 1 indicates that there is an edge between $i-1$ and $i$, i.e., the two pixels $i-1$ and $i$ belong to different homogeneous regions. Thus, we have
$$K_i^l = 0.$$
To obtain $K_i^r$, we need to compare $|u_j^{(n)} - u_i^{(n)}|$ with the threshold parameter $T$, for $j \in \mathcal{N}_k^r(i)$. If $|u_j^{(n)} - u_i^{(n)}| \le T$ for all $j$, then $K_i^r = k$. That is, all pixels in $\mathcal{N}_k^r(i)$ belong to $\Omega_k^r(i)$; which is presented in Fig. 2(a1). The corresponding weighted average scheme is given by
$$u_i^{(n+1)} = v_4, \quad \text{for } K_i^l = 0,\ K_i^r = k.$$
If $K_i^r < k$, then $|u_i^{(n)} - u_j^{(n)}| \le T$, for all $i < j \le i + K_i^r$, and $|u_i^{(n)} - u_{i+(K_i^r+1)}^{(n)}| > T$. It means that the pixel $i+(K_i^r+1)$ belongs to another homogeneous region;

see Fig. 2(a2). Thus, we assume that there exists a Neumann boundary condition at $i + K_i^r$, to prevent the diffusion between the two regions, and $u_i^{(n+1)}$ is computed by

$$u_i^{(n+1)} = v_3, \quad \text{for } K_i^l = 0,\ K_i^r < k.$$

Except Case 1, one can find that two pixels $i$ and $i-1$ are in the same homogeneous region for the remaining four cases. Therefore, the homogeneous neighbors of $i-1$ belonging to $\mathcal{N}_k^l(i)$ and $\mathcal{N}_k^r(i)$ are also that of $i$.

- **Case 2**: Since $K_{i-1}^l \geq k-1$, we have $|u_{i-1}^{(n)} - u_j^{(n)}| \leq T$ for $\forall j \geq (i-1) - (k-1)$, i.e., $\forall j \geq i-k$. Consequently, all pixels of $\mathcal{N}_k^l(i)$ are the homogeneous neighbors of $i$, and $\Omega_k^l(i) = \mathcal{N}_k^l(i)$. Obviously, we have

$$K_i^l = k.$$

If $K_{i-1}^r = k$, then $\Omega_k^r(i-1) = \{j : i-1 < j \leq (i-1)+k\}$. Therefore, the pixels from $i+1$ to $i+(k-1)$ are in $\Omega_k^r(i)$. The only thing is to judge whether the pixel $i+k$ belongs to $\Omega_k^r(i)$. The index $K_i^r$ can be divided as follows:

$$K_i^r = \begin{cases} k, & \text{if } |u_i^{(n)} - u_{i+k}^{(n)}| \leq T, \\ k-1, & \text{if } |u_i^{(n)} - u_{i+k}^{(n)}| > T. \end{cases}$$

which correspond to Fig. 2(b1-b2), respectively. Two different weighted average schemes are written as

$$u_i^{(n+1)} = \begin{cases} v_1, & \text{for } K_i^l = k,\ K_i^r = k, \\ v_3, & \text{for } K_i^l = k,\ K_i^r = k-1. \end{cases}$$

Note, we compute $u_i^{(n+1)} = v_3$, because there is an edge between $i+(k-1)$ and $i+k$ from $K_i^r = k-1$.

- **Case 3**: Since $\Omega_k^l(i-1) = \{j : (i-1) - K_{i-1}^l \leq j < i-1\}$ for $K_{i-1}^l < k-1$, it is not difficult to obtain $\Omega_k^l(i) = \{j : i - (K_{i-1}^l + 1) \leq j < i\}$. $K_i^l$ is given by

$$K_i^l = K_{i-1}^l + 1.$$

For $K_{i-1}^r = k$, it is the same as that in Case 2, and thus the index $K_i^r$ is equal to $k$ with $|u_i^{(n)} - u_{i+k}^{(n)}| \leq T$ or $k-1$ with $|u_i^{(n)} - u_{i+k}^{(n)}| > T$. Consequently, two weighted average schemes for two types shown in Fig. 2(c1-c2) are proposed:

$$u_i^{(n+1)} = \begin{cases} v_2, & \text{for } K_i^l = K_{i-1}^l + 1,\ K_i^r = k, \\ v_4, & \text{for } K_i^l = K_{i-1}^l + 1,\ K_i^r = k-1. \end{cases}$$

Note, we assume two Neumann boundary conditions at both $i - K_i^l$ and $i + K_i^r$ when $K_i^l = K_{i-1}^l + 1$ and $K_i^r = k-1$.

- **Case 4**: By the similar discussion on $K_i^l$ in Case 2, we also have $K_i^l = k$. From $0 < K_{i-1}^r < k$, there is an edge between $(i-1)+K_{i-1}^r$ and $(i-1)+(K_{i-1}^r+1)$. Accordingly, we have $\Omega_k^r(i) = \{j : i < j \leq i + (K_{i-1}^r - 1)\}$ and $K_i^r$ is obtained by

$$K_i^r = K_{i-1}^r - 1,$$

which is shown in Fig. 2(d). We compute $u_i^{(n+1)}$ by

$$u_i^{(n+1)} = v_3, \quad \text{for } K_i^l = k,\ K_i^r = K_{i-1}^r - 1.$$

- **Case 5**: In this case, it is easy to obtain $K_i^l = K_{i-1}^l + 1$ and $K_i^r = K_{i-1}^r - 1$; see Fig. 2(e). We assume two Neumann boundary conditions at both $i - K_i^l$ and $i + K_i^r$, and $u^{(n+1)}$ is computed as follows:

$$u_i^{(n+1)} = v_4, \quad \text{for } K_i^l = K_{i-1}^l + 1,\ K_i^r = K_{i-1}^r - 1.$$

**Acknowledgments.** This work was supported by the National Natural Science Foundation of China (NSFC) (Nos. 11871035, 11531013, and 11771420), Recruitment Program of Global Young Expert, and HDU grant KYS075618129. We thank the authors of Wang et.al [43] for providing us their code. We also thank the anonymous referees for valuable comments and suggestions, which significantly improved the content of the paper.

## REFERENCES

[1] P. Blomgren and T. F. Chan, Color TV: total variation methods for restoration of vector-valued images, *IEEE Trans. Image Process.*, **7** (1998), 304–309.
[2] J. E. Boyd and J. Meloche, Binary restoration of thin objects in multidimensional imagery, *IEEE Trans. Pattern Anal. Mach. Intell.*, **20** (1998), 647–651.
[3] C. Brito-Loeza, K. Chen and V. Uc-Cetina, Image denoising using the Gaussian curvature of the image surface, *Numer. Math. Part. D. E.*, **32** (2016), 1066–1089.
[4] A. Buades, B. Coll and J. M. Morel, A review of image denoising algorithms with a new one, *Multiscale Model. Simul.*, **4** (2005), 490–530.
[5] J. Cai, S. Osher and Z. Shen, Split Bregman methods and frame based image restoration, *Multiscale Model. Simul.*, **8** (2009), 337–369.
[6] J. Cai, B. Dong and Z. Shen, Image restoration: A wavelet frame based model for piecewise smooth functions and beyond, *Appl. Comput. Harmon. Anal.*, **41** (2015), 94–138.
[7] J. Cai, H. Ji, Z. Shen and G. Ye, Data-driven tight frame construction and image denoising, *Appl. Comput. Harmon. Anal.*, **37** (2014), 89–105.
[8] R. H. Chan, T. F. Chan, L. Shen and Z. Shen, Wavelet algorithms for high-resolution image reconstruction, *SIAM J. Sci. Comput.*, **24** (2003), 1408–1432.
[9] T. F. Chan, S. Esedoglu and M. Nikolova, Finding the global minimum for binary image restoration, in *IEEE International Conference on Image Processing 2005*, (2005), 75–89.
[10] C. S. Cho and S. Lee, Effective Five Directional Partial Derivatives-based Image Smoothing and a Parallel Structure Design, *IEEE Trans. Image Process.*, **25** (2016), 1617–1625.
[11] R. Choksi, Y. van Gennip and A. Oberman, Anisotropic total variation regularized $L_1$ approximation and denoising/deblurring of 2D bar codes, *Inverse Probl. Imaging*, **5** (2011), 591–617.
[12] N. Chumchob, K. Chen and C. Brito-Loeza, A new variational model for removal of combined additive and multiplicative noise and a fast algorithm for its numerical approximation, *Int. J. Comput. Math.*, **90** (2013), 140–161.
[13] P. Coupé, P. Hellier, C. Kervrann and C. Barillot, Nonlocal means-based speckle filtering for ultrasound images, *IEEE Trans. Image Process.*, **18** (2009), 2221–2229.
[14] K. Dabov, A. Foi, V. Katkovnik and K. Egiazarian, Image denoising by sparse 3-D transform-domain collaborative filtering, *IEEE Trans. Image Process.*, **16** (2007), 2080–2095.
[15] C. A. Deledalle, V. Duval and J. Salmon, Anisotropic non-local means with spatially adaptive patch shapes, in *SSVM: Scale Space and Variational Methods in Computer Vision*, **6667** (2011), 231–242.
[16] C. A. Deledalle, A. Charles, V. Duval and J. Salmon, Non-local Methods with Shape-Adaptive Patches (NLM-SAP), *J. Math. Imaging Vis.*, **43** (2012), 103–120.
[17] F. Dong and Y. Chen, A fractional-order derivative based variational framework for image denoising, *Inverse Probl. Imag.*, **10** (2016), 1066–1089.
[18] D. L. Donoho and J. M. Johnstone, Ideal spatial adaptation by wavelet shrinkage, *Biometrika*, **81** (1994), 425–455.
[19] D. L. Donoho, De-noising by soft-thresholding, *IEEE Trans. Inf. Theory*, **41** (1995), 613–627.
[20] M. Elad and M. Aharon, Image denoising via sparse and redundant representations over learned dictionaries, *IEEE Trans. Image Process.*, **15** (2006), 3736–3745.

[21] M. Figueiredo and R. Nowak, An EM algorithm for wavelet-based image restoration, *IEEE Trans. Image Process.*, **12** (2003), 906–916.
[22] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, $2^{nd}$ edition, Prentice Hall, 2002.
[23] X. Gu, H. Wang and D. Yu, Binary Image Restoration Using Pulse Coupled Neural Network, in *ICNIP*, (2001), 922–927.
[24] W. Guo, J. Qin and V. A. Luminita, A geometry guided image denoising scheme, *Inverse Probl. Imag.*, **7** (2013), 1066–1089.
[25] R. A. Horn and C. R. Johnson, *Matrix Analysis*, Cambridge University Press, 1990.
[26] Y. Huang, M. K. Ng and Y. Wen, A new total variation method for multiplicative noise removal, *SIAM J. Imaging Sci.*, **2** (2009), 20–40.
[27] H. Ji, Y. Luo and Z. Shen, Image recovery via geometrically structured approximation, *Appl. Comput. Harmon. Anal.*, **41** (2016), 75–93.
[28] Z. Jin and X. Yang, A variational model to remove the multiplicative noise in ultrasound images, *J. Math. Imaging Vis.*, **39** (2011), 62–74.
[29] M. Karaman, M. A. Kutay and G. Bozdagi, An adaptive speckle suppression filter for medical ultrasonic imaging, *IEEE Trans. Med. Imag.*, **14** (1995), 283–292.
[30] D. T. Kuan, A. Sawchuk, C. Timothy and P. Chavel, Adaptive noise smoothing filter for images with signal-dependent noise, *IEEE Trans. Pattern Anal. Mach. Intell.*, **7** (1985), 165–177.
[31] J. Lee, Digital image enhancement and noise filtering by use of local statistics, *IEEE Trans. Pattern Anal. Mach. Intell.*, **2** (1980), 165–168.
[32] F. Li, M. K. Ng and C. Shen, Multiplicative noise removal with spatially varying regularization parameters, *SIAM J. Imaging Sci.*, **3** (2010), 1–20.
[33] J. E. Odegard, H. Guo, M. Lang, C. Burrus, R. Wells, L. Novak and M. Margarita, Wavelet based SAR speckle reduction and image compression, in *SPIE*, (1995), 17–21.
[34] S. Ono, K. Morinaga and S. Nakayama, Two-dimensional barcode decoration based on real-coded genetic algorithm, in *IEEE CEC*, (2008), 1068–1073.
[35] P. Perona and J. Malik, Scale-space and edge detection using anisotropic diffusion, *IEEE Trans. Pattern Anal. Mach. Intell.*, **12** (1990), 629–639.
[36] S. V. Richard, *Matrix Iterative Analysis*, $2^{nd}$ edition, Springer-Verlag, New York, 2009.
[37] L. Rudin, S. Osher and E. Fatemi, Nonlinear total variation based noise removal algorithms, *Phys. D, Nonlinear Phenomena*, **60** (1992), 259–268.
[38] Y. Shen, B. Han and E. Braverman, Adaptive frame-based color image denoising, *Appl. Comput. Harmon. Anal.*, **41** (2015), 54–74.
[39] Y. Shen, E. Y. Lam and N. Wong, A signomial programming approach for binary image restoration by penalized least squares, *IEEE Trans. Circuits Sys. II: Exp. Briefs*, **55** (2008), 41–45.
[40] C. Sheng, Y. Xin, L. Yao and S. Kun, Total variation-based speckle reduction using multi-grid algorithm for ultrasound images, in *ICIAP*, (2005), 245–252.
[41] M. E. Taylor, *Partial Differential Equations I: Basic Theory*, $2^{nd}$ edition, Springer-Verlag, New York, 2011.
[42] C. Tomasi and R. Manduchi, Bilateral filtering for gray and color images, in *ICCV*, (1998), 839–846.
[43] S. Wang, T. Huang, X. Zhao, J. Mei and J. Huang, Speckle noise removal in ultrasound images by first- and second-order total variation, *Numer. Algor.*, **78** (2018), 513–533.
[44] W. Wang, C. Wu and J. Deng, A general selective averaging method for piecewise constant signal and image processing, *J. Sci. Comput.*, **76** (2018), 1078–1104.
[45] W. Wang, S. Wen, C. Wu and J. Deng, Denoising piecewise constant images with selective averaging and outlier removal, *Numer. Math. Theor. Meth. Appl.*, **12** (2019), 467–491.
[46] C. Wu and X. Tai, Augmented Lagrangian Method, Dual Methods, and Split Bregman Iteration for ROF, Vectorial TV, and High Order Models, *SIAM J. Imaging Sci.*, **3** (2010), 300–339.
[47] L. Xu, C. Lu, Y. Xu and J. Jia, Image smoothing via $L_0$ gradient minimization, *ACM Trans. Graph.*, **30** (2011), 174:1–174:12.
[48] J. Yu, J. Tan and Y. Wang, Ultrasound speckle reduction by a SUSAN-controlled anisotropic diffusion method, *Pattern Recogn.*, **43** (2010), 3083–3092.
[49] J. Zhang and K. Chen, A total fractional-order variation model for image restoration with non-homogeneous boundary conditions and its numerical solution, *SIAM J. Imaging Sci.*, **8** (2015), 2487–2518.

[50] X. Zhao, F. Wang and M. K. Ng, A new convex optimization model for multiplicative noise and blur removal, *SIAM J. Imaging Sci.*, **7** (2014), 456–475. arXiv:1809.09783v2 arXiv:1809.03948v1

Received March 2018; revised March 2019.

*E-mail address:* wnwang@hdu.edu.cn
*E-mail address:* wucl@nankai.edu.cn
*E-mail address:* dengjs@ustc.edu.cn