Swarm Intelligence for Additive White Gaussian Noise Level Estimation

Heri Prasetyo and Umi Salamah
Department of Informatics, Universitas Sebelas Maret (UNS), Surakarta, Indonesia

Abstract
This paper presents a simple technique for estimating the noise levels in noisy images corrupted by additive white Gaussian noise. The proposed technique modifies the existing singular-value-decomposition-based noise level estimation method. The proposed method calculates the sum of trailing singular values to infer noise levels. Particle swarm optimization and its variants can be used compute the optimal scalar value for the proposed noise level estimation method over a set of training images. As discussed in the experimental section, the proposed method outperforms existing schemes on noise level estimation tasks. Additionally, the estimated noise obtained from the proposed method can be used to improve the quality of denoised images.

Keywords: Gaussian, Noise estimation, Scalar value, Singular value, Swarm intelligence

1. Introduction
Numerous attempts have been made to improve the performance of particle swarm optimization (PSO) and its variants. Such methods can be regarded as meta-heuristic algorithms with population-based approaches. These algorithms mimic the behavior of organisms to perform function optimization. The inspiration for PSO and its variants comes from bird movement in the sky, fish schooling in the ocean, and other swarm movements in nature. Candidate solutions in PSO are represented by particles. A PSO population typically consists of many swarm particles and the position of each particle is iteratively updated based on a fitness function. PSO and its variants perform function optimization using a derivative-free approach. One does not typically need to know the derivative of a given function to perform optimization. Overall, PSO and its variants have yielded many promising results in the scientific research and engineering fields.

Additionally, noise level estimation and image denoising algorithms have also attracted significant attention in recent years [6–11]. The noise level estimation method presented in [6] aims to infer the level of noise in an image corrupted with additive white Gaussian noise (AWGN). It calculates the sum of trailing singular values in noisy image to estimate noise levels. In general, image denoising [7–11] attempts to remove AWGN from noisy images using various strategies. Many image denoising methods require prior knowledge regarding noise levels to suppress the occurrence of noise. Therefore, one must know the noise level in an image in advance to perform image denoising.

However, image denoising with prior knowledge is impractical in many image processing and computer vision applications. In an ideal situation, noise levels should be directly estimated...
from noisy images. In this paper, we consider an image denoising task without prior knowledge. Therefore, the noise level for image denoising must be directly estimated from a noisy image. We present a modified version of the singular value decomposition (SVD)-based noise level estimation method presented in [6]. The parameters for SVD-based estimation are obtained by exploiting the PSO algorithm. The proposed method overcomes the challenge of image denoising when prior information regarding noise levels is unavailable. The proposed method can be further extended for multiple secret sharing [12, 13], progressive secret sharing [14], vehicle verification [15], and other image processing and computer vision tasks. This remainder of this paper is organized as follows. Section 2 briefly describes the basic process of PSO and its variants. Section 3 details the SVD-based noise level estimation method that was presented in [6]. Limitations of this method are discussed in this section and we present the proposed method for performing noise level estimation. Section 4 reports detailed experimental results. Finally, our conclusions and plans for future work are discussed.

2. PSO and Its Variants

This section discusses the PSO algorithm and its variants. PSO [1] has been successfully applied to various optimization problems in scientific and engineering tasks. Based on its simplicity, various modifications to the basic PSO algorithm have been proposed and analyzed in the literature [2–6]. PSO mimics the behaviors of searching strategies in flocking birds, swarm communities, schooling fish, and other swarms of organisms in nature.

PSO operates on an iterative population. PSO consists of $N$ swarm particles, denoted as $x_i$, for $i = 1, 2, \ldots, N$, where $i$ and $N$ denote the $i$-th particle and the total number of swarm particles, respectively. Each particle consists of $d$ scalar values representing candidate solutions, meaning $x_i = [x_1, x_2, \ldots, x_d]^T$ or $x_i \in \mathbb{R}^d$. Initially, PSO initializes all swarm particles with uniform random numbers in the range $[a, b]$, which are denoted as $x_i \sim U(a, b)$ for $i = 1, 2, \ldots, d$, where $U(a, b)$ is a uniform random number generator in the range $[a, b]$. The positions of all swarm particles are iteratively updated based on their fitness scores. PSO searches for the optimal swarm particle $x^*$ according to the following optimization problem:

$$\min_{x} f(x),$$  \hspace{1cm} (1)

where $f(x)$ represents the fitness function.

In the PSO algorithm and its variants, each particle maintains its position vector $x_i$, velocity vector $v_i$, best position vector $p_i$, and global best position $p_g$. Let $x_i(t)$ and $v_i(t)$ be the position and velocity of the $i$-th swarm particle in the $t$-th iteration (or generation). The position of the $i$-th swarm particle in the $(t+1)$-th iteration is updated according to the following rule:

$$x_i(t + 1) \leftarrow x_i(t) + v_i(t).$$  \hspace{1cm} (2)

The PSO can achieve local or global optimal solutions at the end of iteration. The swarm particle with the best fitness value is regarded as the optimum solution $x^*$.

The most critical step in PSO and its variants is the updating of velocity. Different updating strategies lead to different optimal values. The following subsections describe how a few different PSO variants handle velocity updates.

2.1 Classical PSO

Velocity updating in classical PSO [1] is defined as follow:

$$v_i(t + 1) = w v_i(t) + c_1 r_1 [p_i - x_i(t)] + c_2 r_2 [p_g - x_i(t)],$$  \hspace{1cm} (3)

where $w$ is the inertia weight, and $c_1$ and $c_2$ are the individual and social cognition constants, respectively. The values of $r_1$ and $r_2$ are uniform random numbers, meaning $r_1, r_2 \sim U(0, 1)$. The values of $c_1$, $c_2$, and $w$ are commonly set to two, two, and 0.9, respectively.

2.2 Linearly Decreased Weight PSO

Velocity updating in linearly decreased weight PSO (LDWPSO) [2] is defined as follows:

$$v_i(t + 1) = \left[ w_2 - \frac{t}{T} (w_2 - w_1) \right] v_i(t) + c_1 r_1 [p_i - x_i(t)] + c_2 r_2 [p_g - x_i(t)],$$  \hspace{1cm} (4)

where $w_1$ and $w_2$ denote the lower bound and upper bound of the inertia weight, respectively. It is common to set $w_1$ and $w_2$ to 0.4 and 0.9, respectively. LDWPSO gradually reduces the inertia weight as iteration progresses.

2.3 Constriction PSO

The main goal of CoPSO is to reduce the number of PSO parameters by eliminating the usage of an inertia weight. Velocity
updating in constriction PSO (CoPSO) \[^3\] is defined as follows:

\[
v_i(t+1) = K \{v_i(t) + c_1 r_1 [p_i - x_i(t)] + c_2 r_2 [p_g - x_i(t)] \},
\]

where \( K \) is a constriction parameter (i.e., combination function between \( c_1 \) and \( c_2 \)). The value of \( K \) can be computed as \( \frac{2}{\sqrt{1 - c_1} - \sqrt{2 - \sqrt{1 - c_2^2}} - 4} \), where \( \varphi = c_1 + c_2 > 4 \). If \( c_1 = c_2 = 2.05 \), then we can obtain \( \varphi = 4.1 \) and \( K = 0.729 \).

### 2.4 Absolute Gaussian PSO

Similar to the CoPSO algorithm, Absolute Gaussian PSO (AGPSO) \[^4\] eliminates several predetermined PSO parameters. As reported in \[^4\], the stochastic coefficients for individual and social cognitions should lie in the range of \([0.72, 0.86]\). The absolute value of a Gaussian random number is an ideal solution for the stochastic coefficients. Therefore, velocity updating in AGPSO is simplified as follows:

\[
v_i(t+1) = |N(0,1)| \{p_i - x_i(t)] + |N(0,1)| \{p_g - x_i(t)] ,
\]

where \( |N(0,1)| \) denotes the absolute value of a Gaussian random number with zero mean and unit variance.

### 2.5 Proportional-Integral-Derivative-Based Strategy PSO

Based on the observation in \[^5\], the updated position of each swarm particle depends on the cumulative momentum in the previous iteration. Velocity updating in proportional-integral-derivative-based strategy PSO (PBPSO) \[^5\] can be set to use the same method as classical PSO or its variants. However, the main difference lies in the position updating for each swarm particle. This position updating is defined as follows:

\[
x_i(t+1) = x_i(t) + v_i(t+1) + K_d D_i(t+1),
\]

where \( D_i(t+1) \) and \( K_d \) are the derivative term and its weight, respectively. The derivative term \( D_i(t+1) \) can be calculated as follows:

\[
D_i(t+1) = \gamma D_i(t) + \{p_g(t) - p_g(t-1)\},
\]

where \( \gamma \) is the weight decay. The values of \( \gamma \) and \( K_d \) are typically set to 0.9 and 0.03, respectively.

3. **Swarm-Intelligence-Based Noise Level Estimation**

This section first reviews the traditional SVD-based noise level estimation method \[^6\]. The proposed swarm-intelligence-based noise level estimation method is also presented in this section. We begin by presenting the formal model of AWGN. Let \( F \) be an input image in grayscale. The AWGN process on \( F \) is modeled as

\[
A = F + N(\theta),
\]

where \( A \) denotes the noisy image and \( N(\theta) \) represents the AWGN generated under a Gaussian distribution \( N(\theta) \sim N'(0, \theta^2 I) \). The symbol \( \theta \) represents the noise level of the AWGN. This study aimed to estimate the noise level from \( A \) such that the estimated noise level would be as close as possible to the original input noise level, meaning \( \hat{\theta} \approx \theta \).

#### 3.1 SVD-Based Noise Level Estimation

This section describes noise level estimation based on the information obtained from SVD computation. It was proven in \[^6\] that SVD yields important information related to noise levels. In some extensions, the singular value matrix of a noisy image preserves this noise level information. To infer the noise level of a noisy image \( A \), we first decompose this image using SVD as

\[
A \mapsto U_A \Sigma_A V_A^T,
\]

where \( U_A \) and \( V_A \) are left and right singular vectors satisfying the unitary property as \( U_A U_A^T = U_A^T U_A = I \) and \( V_A V_A^T = V_A^T V_A = I \), respectively. \( \Sigma_A \) is a singular value matrix consisting of non-increasing singular values along its diagonal, which is denoted as \( \Sigma_A = \text{diag} \{\sigma_1^A, \sigma_2^A, \ldots, \sigma_r^A, 0, \ldots, 0\} \), where \( \sigma_1^A \geq \sigma_2^A \geq \cdots \geq \sigma_r^A > 0 \). The symbol \( r \) represents the rank of the matrix \( A \).

The sum of trailing singular values from matrix \( A \) is then defined as

\[
P_A(M) = \frac{1}{M} \sum_{i=r-M+1}^r \sigma_i^A,
\]

where \( P_A(M) \) is a function with a specific parameter \( M \), where \( 1 \leq M \leq r \). This function provides a clue regarding the noise level in \( A \). The value of \( M \) is commonly set to \( M = \frac{3}{4} r \). \[^6\]

It was demonstrated in \[^6\] that the value of \( P_A(M) \) is linearly
dependent on the noise level $\theta$. However, the value of $P_A(M)$ is image independent. The relationship between $P_A(M)$ and $\theta$ can be formulated using a simple linear function as

$$P_A(M) = \alpha \theta,$$  

(12)

where $\alpha$ denotes a specific scalar. The least-squares fitting (LSF) method is a representative approach that can be incorporated to obtain the coefficient $\alpha$. Therefore, the noise level can be simply estimated based on the sum of the learned singular values of $A$. In other words, $P_A(M)$ if one knows the value of $\alpha$.

However, a simple linear function of (12) is not sufficient to estimate the noise level in an image accurately. We must perform additional processes to derive a more accurate estimated noise level. The AWGN process can be applied to a noisy image $A$ as follows:

$$B \leftarrow A + N(\theta_B),$$  

(13)

where $B$ is the noisy version of $A$. Here, $N(\theta_B)$ denotes the AWGN process with a noise level $\theta_B$, where $N(\theta_B) \sim \mathcal{N}[0, \theta_B^2]$. Applying SVD to $B$ yields the following result:

$$B \rightarrow U_B \Sigma_B V_B^T,$$  

(14)

where $U_A$ and $U_B$ are two unitary matrices, meaning $U_B U_B^T = U_B^T U_B = I$ and $V_B V_B^T = V_B^T V_B = I$. The symbol $\Sigma_B$ represents a singular value matrix, meaning $\Sigma_B = \text{diag}\{\sigma_1^B, \sigma_2^B, \ldots, \sigma_r^B, 0, \ldots, 0\}$.

Similar to the previous discussion, the trailing singular values of $B$ are formally defined as follows:

$$P_B(M) = \frac{1}{M} \sum_{i=r-M+1}^{r} \sigma_i^B,$$  

(15)

where $P_B(M)$ is a function of the parameter $M$. A simple linear function based on (12) is not sufficiently accurate for fitting the function $P_B(M)$. Therefore, the remedial fitting of $P_A(M)$ and $P_B(M)$ is performed using a simple linear function over two scalar values. A more suitable linear function for $P_A(M)$ and $P_B(M)$ can then be defined as follows [6]:

$$\begin{cases} P_A(M) = \alpha \theta + \beta, \\ P_B(M) = \alpha \sqrt{\theta^2 + \theta_B^2} + \beta, \end{cases}$$  

(16)

where $\alpha$ and $\beta$ are two scalar values of the linear function. The values of $\alpha$ and $\beta$ are noise-dependent and image-dependent parameters, respectively. By using the algebraic substitution method, the two equations in (16) can be modified as follows:

$$P_A(M) - \alpha \theta = P_B(M) - \alpha \sqrt{\theta^2 + \theta_B^2}. $$

Using simple algebraic manipulation, we obtain the following forms:

$$\alpha \sqrt{\theta^2 + \theta_B^2} - \alpha \theta = P_B(M) - P_A(M),$$

$$\left\{ \frac{1}{\alpha} \{P_B(M) - P_A(M)\} + \theta \right\}^2 = \theta^2 + \theta_B^2, $$

$$\frac{2\theta}{\alpha} \{P_B(M) - P_A(M)\} + \theta^2 + \frac{1}{\alpha^2} \{P_B(M) - P_A(M)\}^2 = \theta^2 + \theta_B^2. $$

(17)

The formulation in (17) can be further simplified as follows:

$$\frac{2\theta}{\alpha} \{P_B(M) - P_A(M)\} = \theta_B^2 - \frac{1}{\alpha^2} \{P_B(M) - P_A(M)\}^2,$$

$$\frac{2\theta}{\alpha} = \frac{\theta_B^2}{[P_B(M) - P_A(M)]^2} - \frac{1}{\alpha^2} \{P_B(M) - P_A(M)\}. $$

(18)

Finally, the value of $\theta$ can be obtained as follows:

$$\hat{\theta}_{SVD} = \frac{\alpha \theta_B^2}{2 \{P_B(M) - P_A(M)\}} - \frac{P_B(M) - P_A(M)}{2\alpha}, $$

(19)

where $\hat{\theta}_{SVD}$ denotes the estimated noise level obtained from the SVD approach. Based on this algebraic manipulation, one can see that the noise level can be simply estimated based on the learned singular values $A$ and $B$ as $P_A(M)$ and $P_B(M)$, respectively. However, such noise level estimation requires a scalar value of $\alpha$. In a previous scheme [6], the value of $\alpha$ was simply computed by applying the LSF method in (12) to a set of training images, where different image sizes yield different scalar values.

### 3.2 PSO-Based Noise Level Estimation

The most critical step in SVD-based noise level estimation is the determination of $\alpha$. Based on our observations, the LSF method is not sufficient for estimating the value of $\alpha$ accurately. In the proposed method, we determine the value of $\alpha$ with help from the PSO algorithm and its variants. These algorithms require a set of training images. Let $S_A$ be a set of training images...
consisting of $T$ images with a fixed predetermined image size. The image set $S_A$ is denoted as

$$S_A = \{A_1, A_2, \ldots, A_T\}, \quad (20)$$

where $A_t$ is the $t$-th training image corrupted by AWGN for $t = 1, 2, \ldots, T$. The symbol $T$ denotes the number of training images used in the computation of $\alpha$. All images in $S_A$ should have the same image size because the computation of trailing singular values depends on the image size. The rank of the matrix is also dependent on the image size.

All images in $S_A$ are corrupted by AWGN. The noise level of the AWGN in $S_A$ should be determined and recorded in advance. The noise level of the AWGN in $S_A$ is recorded as follows:

$$\theta_{GT} = \{\theta_1, \theta_2, \ldots, \theta_T\}. \quad (21)$$

where $\theta_{GT}$ and $\theta_t$ are the ground truth noise level and estimated noise level in the $t$-th image, respectively, for $t = 1, 2, \ldots, T$. Each noisy image $A_t$ corresponds to a noise level $\theta_t$. A noisy image $A_t$ is obtained by injecting AWGN into $A$ with a noise level $\theta_t$.

The PSO algorithms aim to compute the optimal $\alpha$ for a set of training images $S_A$ and ground-truth noise level $\theta_{GT}$. Suppose that we utilize $N$ swarm particles. Here, each particle is considered as a candidate solution (i.e., $\alpha$ value). The initial solution for PSO is randomly generated as $x_i = \alpha \sim U(a, b)$ for $i = 1, 2, \ldots, N$, where $x_i$ denotes the $i$-th swarm particle. The symbol $U(a, b)$ denotes a uniform random number in the range of $[a, b]$. In practice, the values of $a$ and $b$ are related to the image size. Therefore, these two values should be carefully determined in the initial execution of PSO.

The most important component of the PSO algorithm is the fitness function. The different candidate solutions recorded by each swarm particle yield different fitness scores. For noise level estimation, the fitness function for PSO is defined as the mean squared error between the real noise level and estimated noise level. This fitness function is formally defined as follows:

$$\eta(x_i|S_A, \theta_{GT}) = \frac{1}{T} \sum_{t=1}^{T} \left( \theta_t - \hat{\theta}_t | x_i \right)^2, \quad (22)$$

where $\eta(x_i|S_A, \theta_{GT})$ is the fitness value of the $i$-th swarm particle. The values $\theta_t$ and $\hat{\theta}_t$ denote the real (ground truth) noise level and estimated noise level, respectively. For each swarm particle, we estimate the noise level $\hat{\theta}_t$ for each image in $A_t$ based on the value of $\alpha = x_i$. Noise level estimation was performed using the aforementioned SVD-based method.

The optimization problem for noise level estimation can be further defined as follows:

$$\min_{x^*} \eta(x_i|S_A, \theta_{GT}) \quad (23)$$

for $i = 1, 2, \ldots, N$, where $x^*$ is the optimal solution obtained at the end of PSO. This optimal solution is considered as the optimum scalar value of $\alpha$, meaning $\alpha^* = x^*$. Therefore, the estimated noise level can be calculated simply using our proposed method as follows:

$$\hat{\theta}_{PSO} = \frac{\alpha^* \theta_{GT}^2}{2[P_B(M) - P_A(M)]} - \frac{P_B(M) - P_A(M)}{2\alpha^*}, \quad (24)$$

where $\hat{\theta}_{PSO}$ and $\alpha^*$ are the estimated noise level and optimal scalar value obtained from the PSO method, respectively. The computation in (24) is almost identical to that in (19), except for the scalar value $\alpha$. This estimated noise level can be incorporated into image denoising tasks, as illustrated in Figure 1. This figure presents an image denoising process based on the estimated noise level obtained from the proposed method.

4. Experimental Results

This section reports extensive experimental results for the proposed method. We first investigated the performance of swarm intelligence for finding the optimal value of the fitness function. Subsequently, the performance of the proposed noise level estimation method was evaluated in terms of objective measurements. Finally, we examined the performance of the proposed noise level estimation method for image denoising tasks.
4.1 Performance of Swarm Intelligence

The performances of various swarm intelligence methods are documented in this subsection. In our experiments, we considered the six grayscale images presented in Figure 2 as a training set. These images were extracted from the Kodak image dataset. For training purposes, we generated three training sets consisting of all images with various image sizes. The first to third training sets were composed of images with sizes of $128 \times 128$, $256 \times 256$, and $512 \times 512$, respectively. One training set only contained one image size (i.e., all images in the first training set were all of size $128 \times 128$). All images in each training set were corrupted with AWGN at various noise levels, meaning $\theta_{GT} = \{10, 20, \ldots, 100\}$. Figure 3 presents the relationships between $P_A(M)$ and $\theta_{GT}$ for various image sizes. This figure confirms that the noise level $\theta$ can be considered as a simple linear function with $P_A(M)$, as indicated in (12). Subsequently, the noise levels were estimated for all images in each training set. Here, we conducted an additional AWGN process to obtain $B$ with $\theta_B = 50$.

The main goal of swarm intelligence algorithms is to obtain an optimal scalar value $\alpha^*$. Here, we set the number of swarm particles and maximum number of iterations to 50 and 300, respectively, for each PSO algorithm and its variants. Each particle was first initialized to $x_i = U(-50, 50)$, indicating that the minimum and maximum particle positions were $-50$ and 50, respectively. The maximum allowable velocity for each particle swarm was set to 5 to avoid premature convergence.

For the sake of fair comparison, each PSO algorithm was executed in 30 independent runs. The best run, worst run, and other criteria were investigated for these independent runs. Figure 4 presents the convergence histories for the best and worst runs for various PSO variants when the image size is $128 \times 128$. As shown in this figure, PSO and its variants are able to obtain the optimal values indicated by the convergence results after several iterations.

An additional experiment was conducted to analyze the performance of the PSO algorithms in terms of solution quality. We compared the minimum, maximum, mean, and median values, as well as the standard deviations of all PSO algorithms for various image sizes. We computed these values based on all of the independent runs. Table 1 presents the performance comparisons. All of the PSO methods yield almost identical results, but with slight differences in their standard deviation. All PSO variants yield acceptable optimal values with small standard deviations for all image sizes.

The optimal scalar value $\alpha^*$ was also analyzed using various methods. The value of $\alpha^*$ is different for different image sizes because the computation of $P_A(M)$ and $P_B(M)$ depend on the matrix rank, meaning $M = \frac{3}{4}r$. Table 2 lists the optimal values of $\alpha^*$ for various methods. The symbol $\alpha_{LSF}^*$ denotes the optimal $\alpha$ obtained from LSF for the training image set, as formulated in (12). $\alpha_{PSO}^*$ is the optimal $\alpha$ obtained from PSO and its variants. As shown in Table 2, all the values of $\alpha_{PSO}^*$ are identical, indicating that all of the PSO methods yield the same optimal solution. Therefore, the proposed noise level estimation can be considered as a promising candidate for performing noise level estimation.

4.2 Performance of Noise Level Estimation

We further analyzed the performance of the proposed noise level estimation method in terms of objective measurements.

Figure 2. Set of testing images used in our experiments.

Figure 3. Relationships between $P_A(M)$ and the noise level $\theta$ for various image sizes.

Figure 4. Convergence histories for the best and worst runs for various PSO variants when the image size is $128 \times 128$. As shown in this figure, PSO and its variants are able to obtain the optimal values indicated by the convergence results after several iterations.
We evaluated its performance on six grayscale images acting as a testing set. The testing images were extracted from the Kodak image dataset. Figure 5 presents all of the images used in our experiments. In our experiments, all of the images were rescaled to derive sets of testing images of various sizes (e.g., 128×128, 256×256, and 512×512). All of the images were then corrupted using the AWGN process with various noise levels, meaning $\theta_{GT} = \{10, 20, \ldots, 100\}$. We subsequently performed noise level estimation using various methods to obtain an estimated noise level $\hat{\theta}$.

The similarity between the ground truth $\theta_t$ and estimated noise level $\hat{\theta}$ was measured in terms of mean absolute error (MAE), which is as formulated follows:

$$\text{MAE} = \frac{1}{T} \sum_{t=1}^{T} \left| \theta_t - \hat{\theta} \right|. \quad (25)$$

A smaller MAE value indicates better performance for noise level estimation. Table 3 presents performance comparisons in terms of MAE for various methods and image sizes. In our experiments, we applied the AWGN process over 30 times for each noise level and testing image. Therefore, the MAE values in Table 3 represent average values for 30 runs of noise level estimation. The proposed method provides the best performance for all image sizes, as indicated by its small MAE values.

Additionally, the performance of the proposed method was also compared to those of the other methods in terms of stan-
Table 3. Performance comparisons in terms of MAE between the proposed method and previous method

| Noise level | 128 × 128 | | | 256 × 256 | | | 512 × 512 | | |
|-------------|-----------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|
|             | $\alpha^*_{LSF}$ | $\alpha^*[6]$ | $\alpha^*_{PSO}$ | $\alpha^*_{LSF}$ | $\alpha^*[6]$ | $\alpha^*_{PSO}$ | $\alpha^*_{LSF}$ | $\alpha^*[6]$ | $\alpha^*_{PSO}$ |
| 10          | 4.10      | 2.00            | 1.40            | 3.19            | 1.41            | 1.18            | 2.82            | 1.27            | 1.02            |
| 20          | 3.85      | 1.56            | 0.99            | 3.13            | 1.06            | 0.83            | 2.79            | 0.96            | 0.76            |
| 30          | 4.26      | 1.74            | 1.18            | 3.41            | 1.05            | 0.76            | 2.93            | 0.89            | 0.68            |
| 40          | 4.64      | 2.07            | 1.27            | 3.60            | 1.19            | 0.75            | 3.14            | 0.90            | 0.65            |
| 50          | 5.22      | 2.34            | 1.68            | 4.10            | 1.29            | 0.95            | 3.47            | 0.98            | 0.70            |
| 60          | 5.42      | 2.94            | 2.20            | 4.28            | 1.39            | 1.10            | 3.85            | 1.02            | 0.71            |
| 70          | 6.66      | 3.28            | 2.81            | 4.83            | 1.60            | 1.35            | 4.14            | 1.09            | 0.88            |
| 80          | 7.06      | 4.29            | 3.43            | 5.41            | 2.07            | 1.64            | 4.51            | 1.23            | 0.95            |
| 90          | 8.26      | 4.98            | 4.25            | 6.17            | 2.48            | 2.04            | 4.71            | 1.56            | 1.08            |
| 100         | 8.39      | 4.87            | 5.09            | 6.29            | 2.83            | 2.55            | 5.17            | 1.56            | 1.36            |
| Average     | 5.79      | 3.01            | 2.43            | 4.44            | 1.64            | 1.31            | 3.75            | 1.15            | 0.88            |

Table 4. Performance comparisons in terms of standard deviation between the proposed method and previous method

| Noise level | 128 × 128 | | | 256 × 256 | | | 512 × 512 | | |
|-------------|-----------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|
|             | $\alpha^*_{LSF}$ | $\alpha^*[6]$ | $\alpha^*_{PSO}$ | $\alpha^*_{LSF}$ | $\alpha^*[6]$ | $\alpha^*_{PSO}$ | $\alpha^*_{LSF}$ | $\alpha^*[6]$ | $\alpha^*_{PSO}$ |
| 10          | 1.66      | 1.42            | 0.79            | 1.34            | 1.05            | 0.73            | 1.46            | 1.18            | 0.99            |
| 20          | 1.26      | 1.10            | 0.68            | 1.04            | 0.81            | 0.53            | 1.05            | 0.85            | 0.67            |
| 30          | 1.36      | 1.15            | 0.80            | 0.95            | 0.74            | 0.48            | 0.94            | 0.77            | 0.54            |
| 40          | 1.65      | 1.46            | 0.96            | 1.02            | 0.81            | 0.55            | 0.84            | 0.70            | 0.52            |
| 50          | 2.20      | 1.71            | 1.20            | 1.22            | 0.90            | 0.67            | 0.88            | 0.71            | 0.50            |
| 60          | 2.76      | 2.19            | 1.67            | 1.44            | 1.08            | 0.80            | 0.92            | 0.76            | 0.51            |
| 70          | 3.59      | 2.37            | 2.08            | 1.57            | 1.25            | 0.92            | 1.03            | 0.77            | 0.63            |
| 80          | 4.01      | 3.08            | 2.28            | 2.28            | 1.51            | 1.22            | 1.29            | 0.86            | 0.71            |
| 90          | 5.07      | 3.72            | 3.69            | 2.81            | 1.87            | 1.54            | 1.57            | 1.10            | 0.88            |
| 100         | 5.36      | 4.57            | 3.75            | 2.97            | 1.99            | 2.07            | 1.71            | 1.11            | 0.96            |
| Average     | 2.89      | 2.28            | 1.79            | 1.66            | 1.20            | 0.95            | 1.17            | 0.88            | 0.69            |

dard deviation. A smaller standard deviation indicates better stability for noise level estimation. All images in the testing set were corrupted with AWGN 30 times for each noise level and the standard deviations were computed for all testing images. Table 3 presents performance comparisons in terms of standard deviation. One can see that the proposed method yields the best performance in terms of standard deviation for all image sizes. One can conclude that the proposed swarm-intelligence-based noise level estimation offers promising results for estimating AWGN in noisy images.

4.3 Effects of Estimated Noise Level on Image Denoising

We also conducted an experiment to investigate the performance of the proposed noise level estimation method for image denoising tasks. We first discuss its performance based on visual inspections. We consider one image from Figure 5 as a testing image. Figure 5(a) presents a noisy image corrupted with AWGN over $\theta = 55$. It is challenging for the proposed method and other methods to estimate this noise level because it was not included in the training set. Figure 5(b) presents the denoised image obtained from Gaussian filtering, while Figs. 5(c-e) present denoised images after applying the BM3D method [9] with estimated noise levels of $\hat{\theta}_{LSF}$, $\hat{\theta}_{SVD}$, and $\hat{\theta}_{PSO}$, respectively. Figure 5(f) presents the ground-truth image. One can see that the proposed method yields the best performance based on the enhanced visual clarity in the resulting image compared to the other methods.
Another experiment was conducted to measure the performance of the proposed method compared to the other methods in terms of objective measurements. Two metrics were considered in this experiment: peak-signal-to-noise ratio (PSNR) and the structural similarity index metric (SSIM). Higher values of PSNR and SSIM indicate better performance. In this experiment, all of the images in Figure 5 were considered as a testing set. All of the images were rescaled to various sizes (e.g., $128 \times 128$, $256 \times 256$, and $512 \times 512$). All images were corrupted with AWGN at various noise levels of $\theta = 25, 55, 75$. These noise levels were not included in the training set. All noise levels were applied 30 times to all testing images. Average PSNR and SSIM values were computed for all independent runs. Tables 5 and 6 summarize the average PSNR and SSIM values of various methods. One can see that the proposed method yields the best performance in terms of PSNR and SSIM. Therefore, the proposed method is suitable for performing noise level estimation on AWGN-corrupted images.

5. Conclusion

A simple technique for estimating noise levels was presented in this paper. This technique leverages the effectiveness of SVD-based noise level estimation. Noise levels are estimated from noisy images simply by analyzing their trailing singular values. The proposed method exploits a swarm intelligence approach to determine the optimal scalar value for noise level estimation. As documented in the experimental section, the proposed method outperforms previous methods on noise level estimation tasks. In future work, the proposed method can be extended to color images and videos. Additional strategies using PSO and its variants can also be applied to achieve better accuracy for determining the optimal scalar value. Other nature-inspired algorithms can also be investigated to achieve better performance compared to PSO and its variants. The second step in the proposed method of injecting additional AWGN into noisy images could be removed to avoid obscuring important information in images, which would lead to more accurate estimated noise levels.

Conflicts of Interest

No potential conflict of interest relevant to this article was reported.
Figure 6. Image denoising results for (a) a noisy image \( \theta = 55 \) using various methods: (b) Gaussian filtering \( 11 \times 11 \) and \( 1.5 \), (c) BM3D with \( \hat{\theta}_{LSF} \), (d) BM3D with \( \hat{\theta}_{SVD} \), and (e) BM3D with \( \hat{\theta}_{PSO} \). (f) Ground-truth image.

Table 5. Performance comparisons in terms of average PSNR for various noise estimation methods

| Method      | \( \theta = 25 \) | \( \theta = 55 \) | \( \theta = 75 \) |
|-------------|-------------------|-------------------|-------------------|
|             | 128 \times 128   | 256 \times 256   | 512 \times 512   |
| Gaussian filtering | 24.55  | 25.09  | 25.10  | 23.05  | 23.38  | 23.41  | 21.80  | 22.18  | 22.17  |
| \( \hat{\theta}_{LSF} \)   | 27.26  | 28.20  | 28.78  | 24.09  | 24.88  | 25.52  | 22.85  | 23.81  | 24.40  |
| \( \hat{\theta}_{SVD} \) \[6\] | 27.51  | 28.41  | 28.98  | 24.21  | 24.94  | 25.60  | 22.89  | 23.85  | 24.45  |
| \( \hat{\theta}_{PSO} \)   | 27.65  | 28.55  | 24.28  | 25.02  | 25.63  | 22.96  | 23.88  | 24.45  |        |

Table 6. Performance comparisons in terms of average SSIM for various noise estimation methods

| Method      | \( \theta = 25 \) | \( \theta = 55 \) | \( \theta = 75 \) |
|-------------|-------------------|-------------------|-------------------|
|             | 128 \times 128   | 256 \times 256   | 512 \times 512   |
| Gaussian filtering | 0.68   | 0.69   | 0.68   | 0.57   | 0.56   | 0.54   | 0.49   | 0.48   | 0.46   |
| \( \hat{\theta}_{LSF} \)   | 0.78   | 0.80   | 0.80   | 0.65   | 0.67   | 0.69   | 0.58   | 0.63   | 0.64   |
| \( \hat{\theta}_{SVD} \) \[6\] | 0.80   | 0.80   | 0.81   | 0.66   | 0.67   | 0.69   | 0.59   | 0.63   | 0.64   |
| \( \hat{\theta}_{PSO} \)   | 0.80   | 0.81   | 0.81   | 0.66   | 0.68   | 0.69   | 0.59   | 0.63   | 0.64   |
References

[1] R. C. Eberhart, Y. Shi, and J. Kennedy, *Swarm Intelligence*. San Diego, CA: Academic Press, 2001.

[2] Y. Shi and R. C. Eberhart, “Empirical study of particle swarm optimization,” in *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99* (Cat. No. 99TH8406), Washington, DC, 1999, pp. 1945-1950. https://doi.org/10.1109/CEC.1999.785511

[3] M. Clerc and J. Kennedy, “The particle swarm-explosion, stability, and convergence in a multidimensional complex space,” *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 1, pp. 58-73, 2002. https://doi.org/10.1109/4235.985892

[4] R. A. Krohling and L. dos Santos Coelho, “Co-evolutionary particle swarm optimization using Gaussian distribution for solving constrained optimization problems,” *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 36, no. 6, pp. 1407-1416, 2006. https://doi.org/10.1109/TSMCB.2006.873185

[5] Z. Xiang, D. Ji, H. Zhang, H. Wu, and Y. Li, “A simple PID-based strategy for particle swarm optimization algorithm,” *Information Sciences*, vol. 502, pp. 558-574, 2019. https://doi.org/10.1016/j.ins.2019.06.042

[6] W. Liu and W. Lin, “Additive white Gaussian noise level estimation in SVD domain for images,” *IEEE Transactions on Image Processing*, vol. 22, no. 3, pp. 872-883, 2013. https://doi.org/10.1109/TIP.2012.2219544

[7] Q. Guo, C. Zhang, Y. Zhang, and H. Liu, “An efficient SVD-based method for image denoising,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 26, no. 5, pp. 868-880, 2016. https://doi.org/10.1109/TCSVT.2015.2416631

[8] Y. M. Huang, H. Y. Yan, Y. W. Wen, and X. Yang, “Rank minimization with applications to image noise removal,” *Information Sciences*, vol. 429, pp. 147-163, 2018. https://doi.org/10.1016/j.ins.2017.10.047

[9] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, “Image denoising by sparse 3-D transform-domain collaborative filtering,” *IEEE Transactions on Image Processing*, vol. 16, no. 8, pp. 2080-2095, 2007. https://doi.org/10.1109/TIP.2007.901238

[10] S. Gu, L. Zhang, W. Zuo, and X. Feng, “Weighted nuclear norm minimization with application to image denoising,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Columbus, OH, 2014, pp. 2862-2869. https://doi.org/10.1109/cvpr.2014.366

[11] H. Hu, J. Froment, and Q. Liu, “A note on patch-based low-rank minimization for fast image denoising,” *Journal of Visual Communication and Image Representation*, vol. 50, pp. 100-110, 2018. https://doi.org/10.1016/j.jvcir.2017.11.013

[12] H. Prasetyo and J. M. Guo, “A note on multiple secret sharing using Chinese remainder theorem and exclusive-OR,” *IEEE Access*, vol. 7, pp. 37473-37497, 2019. https://doi.org/10.1109/access.2019.2902853

[13] H. Prasetyo and C. H. Hsia, “Improved multiple secret sharing using generalized chaotic image scrambling,” *Multimedia Tools and Applications*, vol. 78, no. 20, pp. 29089-29120, 2019. https://doi.org/10.1007/s11042-018-6304-y

[14] H. Prasetyo and C. H. Hsia, “Lossless progressive secret sharing for grayscale and color images,” *Multimedia Tools and Applications*, vol. 78, no. 17, pp. 24837-24862, 2019. https://doi.org/10.1007/s11042-019-7710-5

[15] J. M. Guo, H. Prasetyo, M. E. Farfoua, and H. Lee, “Vehicle verification using features from curvelet transform and generalized Gaussian distribution modeling,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 4, pp. 1989-1998, 2015. https://doi.org/10.1109/TITS.2014.2386535

Heri Prasetyo received his Bachelor’s Degree from the Department of Informatics Engineering, Institut Teknologi Sepuluh Nopember (ITS), Indonesia in 2006. He received his Master’s and Doctoral Degrees from the Department of Computer Science and Information Engineering, and Department of Electrical Engineering, respectively, at the National Taiwan University of Science and Technology (NTUST), Taiwan, in 2009 and 2015, respectively. He received the Best Dissertation Award from the Taiwan Association for Consumer Electronics (TACE) in 2015, Best Paper Awards from the International Symposium on Electronics and Smart Devices 2017 (ISESD 2017), ISESD 2019, and International Conference on Science in Information Technology (ICSITech 2019), and the...
Outstanding Faculty Award 2019 from his current affiliated university. His research interests include multimedia signal processing, computational intelligence, pattern recognition, and machine learning.

Umi Salamah received her Bachelor’s Degree from the Department of Mathematics, Universitas Sebelas Maret (UNS), Indonesia, in 1994. She received her Master’s and Doctoral Degrees from the Department of Informatics Engineering, Institut Teknologi Sepuluh Nopember (ITS), Indonesia, in 2002 and 2018, respectively. Her research interests include fuzzy logic and systems, image processing, applied mathematics, and computational sciences.