MONTE CARLO OPTIMIZATION OF NON-LOCAL MEANS DENOISING ALGORITHM

ROBU PETRU-RĂZVAN, VERZOTTI MATTEO-ALEXANDRU, VOAIDEȘ-NEGUSTOR ROBERT-IONUȚ

Abstract. Non-Local Means is a powerful algorithm for image denoising, yet its high computational complexity limits real-time application. This paper replicates the Monte Carlo Non-Local Means Optimization, which utilizes random sampling to accelerate weight calculations without significantly degrading peak signal-to-noise ratio.

Keywords. Denoising, Monte Carlo, Optimization

1. Introduction

Image noise is a random variation of brightness or color information. In most real-life cases, it can be modeled as additive white Gaussian noise:

$$y = x + \eta \tag{1.1}$$

Where y is the noisy pixel, x the pure pixel and $\eta \sim \mathcal{N}(0, \sigma)$ is the noise.

1.1. Denoising Techniques.

Traditional denoising techniques, such as Gaussian smoothing, Median filtering or Local means, operate on the principle of locality, assuming that the true value of the pixel must be similar to the values of its neighbours. This is effective at removing noise, but it also leads to loss of fine textures and blurring of edges. Newer techniques include Convolutional Neural Networks, Wavelet Transforms [6], and Modified Decision-Based Median Filter [7]. Although these methods can achieve impressive results, they often require extensive training data, computational resources, or complex parameter tuning. Non-Local Means (NLM) [1] is remarkable in its simplicty and effectiveness, however its high computational complexity limits real-time application.

1.2. Non-Local Means.

The Non-Local Means estimates the true value of the pixel by computing a wieghted average of different patches, using a weight function that prioritizes pixels with similar structural patterns. The most similar pixels to a given noisy pixel have no reason to be in its immediate vicinity. For example, patterns such as smooth surfaces, periodic textures or repeated structures can appear in different areas of the image. Although the search space is larger, we are still going to use only a neighbourhood of the pixel to limit the computational cost. As aknowledged by the authors themselves [2], the term "non-local" is somewhat misleading, and a more accurate name would be *"semi-local"*.

Definition 1.1. Suppose Ω is an image domain, and let p, $q \in \Omega$ be two points within the image. Then, the algorithm for denoising is defined as:

$$z(p) = \frac{1}{C(p)} \int_{\Omega} w(p,q)y(q)\, dq \tag{1.2}$$

where $z(p)$ is the denoised value at point p, $y(q)$ is the noisy value at point q, $w(p,q)$ is the weight function that measures the similarity between patches centered at p and q, and $C(p)$ is a normalizing factor given by:

$$C(p) = \int_{\Omega} w(p,q)\, dq \tag{1.3}$$

Obviously, in a real-world application we would use a discrete version of this algorithm:

$$z(p) = \frac{1}{C(p)} \sum_{q\in\Omega} w(p,q)y(q) \tag{1.4}$$

where, again, the normalizing factor is given by:

$$C(p) = \sum_{q\in\Omega} w(p,q) \tag{1.5}$$

Given a noisy pixel, the surrounding patch $\mathbf{y}$ will be used to check for similarity with other patches. This object can be flattened into a d-pixel: $\mathbf{y} \in \mathbb{R}^d$. The algorithm requires a set of patches $\mathcal{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ that are obtained from reference images. Given this, the discrete version of NLM can be expressed as:

$$z = \frac{\sum_{i=1}^{n} w_i x_i}{\sum_{i=1}^{n} w_i} \tag{1.6}$$

where x_i represents the center pixel in the patch $\mathbf{x_i}$, and the weight w_i measures the similarity between the match $\mathbf{y}$ and $\mathbf{x_i}$.

A standard choice for the weight function is:

$$w_i = e^{-||\mathbf{y}-\mathbf{x_i}||^2/(2h_r^2)} \tag{1.7}$$

Where h_r is a scalar parameter that controls the decay of the distribution: small values create a fast decay and ensure a focus on stricter similarity between patches, whereas larger values allow different patches to have a greater weight. And $||\cdot||$ is the Euclidian norm.

The downside of NLM is it high computational complexity. The most expensive step of this algorithms is computing the weights $\mathbf{w}_i$, which leads to $\mathcal{O}(mnd)$ operations, where m is the number of pixels to be denoised, n the number of patches and d the dimensions of the patch. In reality, a study has shown that it's enough to only use a localized window of dimension D around the pixel to be denoised [5], which reduces the complexity to $\mathcal{O}(mD^2d)$. Keep in mind that m and d are also quadratic in nature of the image and patch sizes.

A fun, but not-so-trivial way of improving the computational complexity is to use summed-area tables and the Fast Fourier Transform, which achieves a theoretical complexity of $\mathcal{O}(mD^2\sqrt{d})$. An implementation of this by Jacques

Froment [4], called *Parameter-Free Fast Pixelwise Non-Local Means Denoising*, claims to speed up the process by a factor of 6 to 49.

In reality, both of these implementations, as well as the one below, are easily parallelizable, and can be sped up even further using GPU computing.

1.3. Monte Carlo Non-Local Means (MCNLM).

This paper focuses on the implementation and effectiveness of a method developed in [3]. The main focus of the approach is to approximate 1.6 by selecting randomly k reference pixel and creating a k-subset of weights $\{w_i\}^k$. The computational complexity of this approach is $\mathcal{O}(mkd)$, which is significantly lower for $k \ll n$.

2. Monte Carlo Non-Local Means

In this section, we discuss the full implementation of the algorithm, the correctness of the approach and some other optimizations that can be added to the method.

2.1. Sampling Patches.

There are two ways to extract reference patches. The first one is based on picking them from the original noisy, image, which is called *internal denoising*. The second method is based on having the patches taken from a large database of other images, which is called *external denoising*. The paper focuses on internal denoising.

The Monte Carlo sampling is done on the set $\mathcal{X} = \{x_1, \ldots, x_n\}$, considering each reference patch independent. The process is determined by a sequence of random variables $\{I_j\}_{j=1}^n$, where $I_j \sim \text{Bernoulli}(p_j)$. The weight will be sampled only if $I_j = 1$. The vector of these probabilities, $\mathbf{p} := [p_1, \ldots, p_n]^T$, is called the *sampling pattern* of the algorithm [3].

The main parameter of this algorithm is ξ, which is the expected value of the random variable which models the ratio between the number of the samples taken and the number of references:

$$S_n = \frac{1}{n}\sum_{j=1}^{n} I_j \tag{2.1}$$

which has:

$$\xi \overset{\text{def}}{=} \mathbb{E}[S_n] = \frac{1}{n}\sum_{j=1}^{n} \mathbb{E}[I_j] = \frac{1}{n}\sum_{j=1}^{n} p_j \tag{2.2}$$

So, an important requirement is that:

$$\sum_{j=1}^{n} = n\xi \tag{2.3}$$

S_n is called the *empirical sampling ratio* and ξ the average sampling ratio [3].

2.2. **Algorithm.** Given a set of references $\mathcal{X}$ and the sampling ration ξ, we can define the sampling pattern $\mathbf{p}$ in order for it to respect the condition:

$$\sum_{j=1}^{n} p_j = n\xi \tag{2.4}$$

.

We can approximate the numerator and the denominator in 1.6 using two random variables:

$$A = \frac{1}{n}\sum_{j=1}^{n} x_j w_j \frac{I_j}{p_j} \quad \text{and} \quad B = \frac{1}{n}\sum_{j=1}^{n} w_j \frac{I_j}{p_j} \tag{2.5}$$

To show why we scale by $\frac{1}{p_j}$, we calculate the expected value of the estimators. By linearity of expectation:

$$\mathbb{E}[A] = \frac{1}{n}\sum_{j=1}^{n} x_j w_j \frac{\mathbb{E}[I_j]}{p_j} = \frac{1}{n}\sum_{j=1}^{n} x_j w_j \frac{p_j}{p_j} = \frac{1}{n}\sum_{j=1}^{n} x_j w_j \tag{2.6}$$

and

$$\mathbb{E}[B] = \frac{1}{n}\sum_{j=1}^{n} w_j \frac{\mathbb{E}[I_j]}{p_j} = \frac{1}{n}\sum_{j=1}^{n} w_j \tag{2.7}$$

So, A and B are *unbiased estimators* of the true numerator and denominator. In the end, we will aproximate the result z by another random variable:

$$Z = \frac{A}{B} = \frac{\sum_{j=1}^{n} x_j w_j \frac{I_j}{p_j}}{\sum_{j=1}^{n} w_j \frac{I_j}{p_j}} \tag{2.8}$$

However,

$$\mathbb{E}[Z] = \mathbb{E}\left[\frac{A}{B}\right] \neq \frac{\mathbb{E}[A]}{\mathbb{E}[B]} = z \tag{2.9}$$

so, Z is a *biased estimator* of z. However, section 3 proves that as $n \to \infty$, the deviation $|Z - z|$ drops exponentially. This shows that, for a larger number of patches, the more efficient MCNLM gives results comparable to that of the simple NLM.

Algorithm 1 Monte Carlo NLM Denoising

```
Require: Noisy patch $\mathbf{y}$, reference set $\mathcal{X}$, sampling pattern $\mathbf{p}$
Ensure: Denoised pixel $z$
1: for $j$ in $1 \dots n$ do
2:     Generate random variable $I_j \sim \text{Bernoulli}(p_j)$
3:     if $I_j = 1$ then
4:         Compute $w_j$
5:     end if
6: end for
7: Compute $A = \frac{1}{n} \sum_{j=1}^{n} w_j x_j \frac{I_j}{p_j}$
8: Compute $B = \frac{1}{n} \sum_{j=1}^{n} w_j \frac{I_j}{p_j}$
9: return $Z = \frac{A}{B}$
```

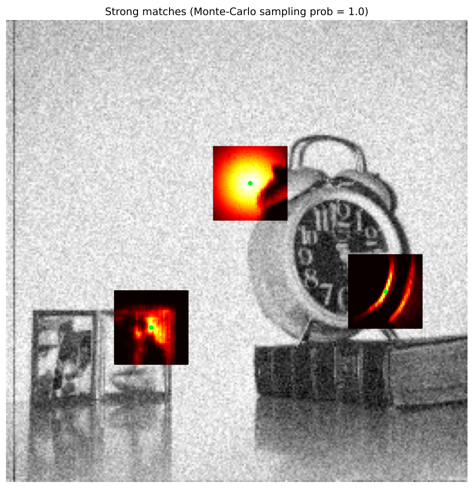
2.3. **Spatial Sampling.** We must also consider that locality matters in images. In a picture of a starry night, for example, a bright star might look like noise if compared to the rest of the dark sky. We can integrate this insight with the ingenious structural handling of MCNLM by incorporating a spatial distance parameter into the weighting function:

$$w_j = w_j^s \cdot w_j^r \tag{2.10}$$

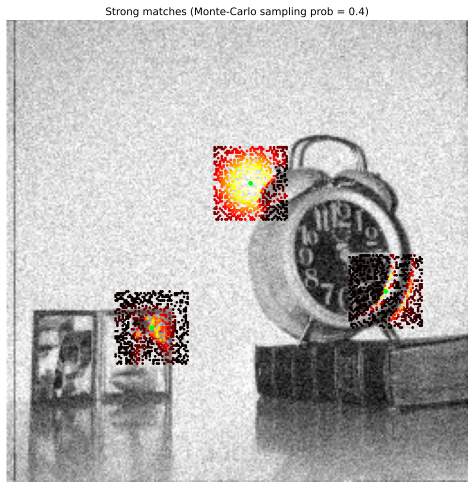
where w_j^r is the weight mentioned at 1.7 and w_j^s is a spatial weight:

$$w_j^s = e^{-(d_2^j)^2/(2h_s^2)} \cdot \mathbb{I}\{d_\infty^j \leq \rho\} \tag{2.11}$$

where d_j^2 is the Euclidean distance between the noisy patch and the reference patch we compare to and d_j^∞ is the Chebyshev distance. The indicator function $\mathbb{I}$ is used to determine a spatial search window of width ρ.



(A) Weight values in search window



(B) Weigths for sampled center pixels

FIGURE 1. Main caption describing both images

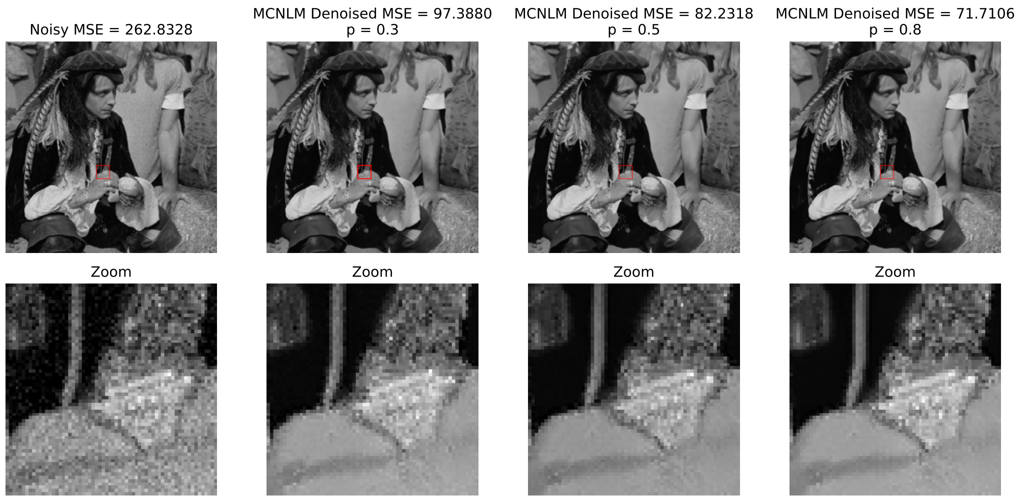

Noisy MSE = 262.8328
MCNLM Denoised MSE = 97.3880
p = 0.3
MCNLM Denoised MSE = 82.2318
p = 0.5
MCNLM Denoised MSE = 71.7106
p = 0.8
Zoom
Zoom
Zoom
Zoom

FIGURE 2. MCNLM with $\xi \in \{0.3, 0.5, 0.8\}$

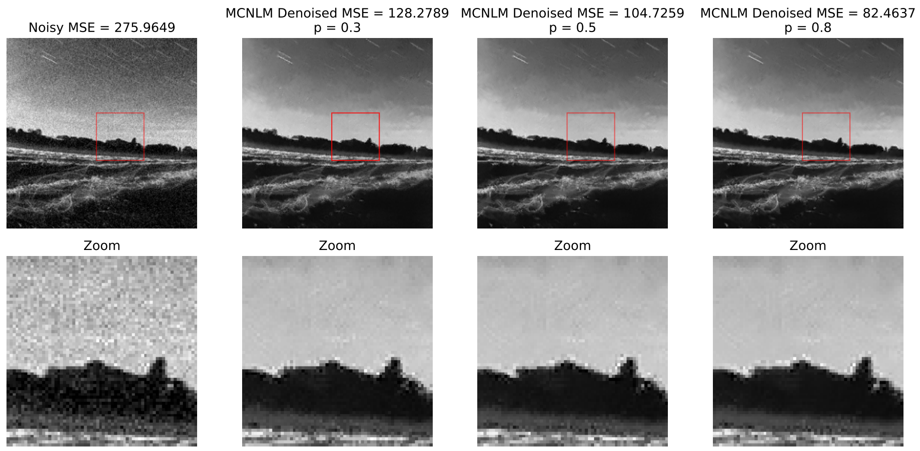

Noisy MSE = 275.9649
MCNLM Denoised MSE = 128.2789
p = 0.3
MCNLM Denoised MSE = 104.7259
p = 0.5
MCNLM Denoised MSE = 82.4637
p = 0.8
Zoom
Zoom
Zoom
Zoom

FIGURE 3. MCNLM with $\xi \in \{0.3, 0.5, 0.8\}$

Figure 4 visualizes the weight function within a search window centered on a noisy pixel, where brighter values indicate higher weight magnitudes. This illustrates how the method preserves structure: pixels with similar patches exert a stronger influence on the weighted average, ensuring that structural details are retained in the denoised output.

2.4. **Results.** In the figures 6, 3 we can see how the MCNLM algorithm behaves on an 1024x1024 image ($\approx 10^6$ pixels), with Gaussian noise with $\sigma = 17/255$ (taken into account that we normalize our pixels to have values in the interval $[0, 1]$)/ and zero mean. The comparison is done on 5x5 patches, and on a 20x20 search window arround the noisy pixel. The sampling pattern is chosen as uniform, with $\mathbf{p} = \{\xi, \cdots, \xi\}$. It is obvious that an increase in the sampling ratio only provides a marginally better result.

3. Stochastic Approximation of Non-Local Means

We yield the question whether the Monte-Carlo variant of our algorithm is indeeed reliable and aproaches the full Non-Local Means solution. For this we need to analyze the approximation error $|Z - z|$.

To understand our approximation, we will study the empirical sampling ratio S_n. The law of large numbers states that the empirical mean S_n of a large number of indepent random variable is close to the true mean.

3.1. Probability Bounds.

3.1. Probability Bounds. For any error bound $\varepsilon > 0$ and any sampling pattern p with $0 < p_j <= 1$ and $\sum_{j=1}^{n} p_j = \varepsilon$ we want to study the probability:

$$\mathbb{P}(|S_n - \mathbb{E}[S_n]| > \varepsilon) \tag{3.1}$$

From the Cebyshev inequality we know that:

$$\mathbb{P}(|S_n - \mathbb{E}[S_n]| > \varepsilon) < \frac{Var[I_1]}{n\varepsilon^2}, \forall \varepsilon > 0 \tag{3.2}$$

This bound provided by the Cebyshev inequality above is too loose, only providing a linear descent of the error probability as $n \to \infty$.

We can use the Bernstein inequality to provide a much tighter exponential bound. The Bernstein inequality states that for $X_1, \dots X_n$, a sequence of independent bounded random variables ($l_j <= X_j <= u_j$, where u_j and l_j are constants), the following is true:

$$\mathbb{P}(|S_n - \mathbb{E}[S_n]| > \varepsilon) \leq exp\left(-\frac{n\varepsilon^2}{2(\frac{1}{n}\sum_{j=1}^{n} Var[X_j] + M\epsilon/3)}\right) \tag{3.3}$$

where $S_n = \frac{1}{n}\sum_{j=1}^{n} X_j$ and $M = max_{1 \leq j \leq n} \frac{u_j - l_j}{2}$
In our situation, $X_j = I_j$, $M = 1$, $\mathbb{E}[S_n] = \xi$, so:

$$\frac{1}{n}\sum_{j=1}^{n} Var[X_j] = \frac{1}{n}\sum_{j=1}^{n} p_j(1 - p_j) = \xi(1 - \xi).$$

By substituing this result in the equation above we get an exponential bound on the error probability.

3.2. General Error Probability Bound. To rigorously quantify the reliability of the Monte Carlo approximation, we refer to the concentration bound derived in the literature. For a sample size n and an error tolerance $\varepsilon > 0$, the probability that the Monte Carlo estimate $Z(\mathbf{p})$ deviates from the exact NLM value z is bounded by exponential decay terms:

$$\begin{aligned}\mathbb{P}(|Z(\boldsymbol{p}) - z| > \varepsilon) \leq & \exp\{-n\xi\} \\ & + \exp\left\{\frac{-n(\mu_B\varepsilon)^2}{2\left(\frac{1}{n}\sum_{j=1}^{n}\alpha_j^2\left(\frac{1-p_j}{p_j}\right) + (\mu_B\varepsilon)M_\alpha/6\right)}\right\} \\ & + \exp\left\{\frac{-n(\mu_B\varepsilon)^2}{2\left(\frac{1}{n}\sum_{j=1}^{n}\beta_j^2\left(\frac{1-p_j}{p_j}\right) + (\mu_B\varepsilon)M_\beta/6\right)}\right\},\end{aligned} \tag{3.4}$$

where:

- ξ represents the sampling density (e.g., p_j in uniform sampling).
- μ_B is the average similarity weight (expected value of the denominator).
- K is a constant bound dependent on the variance of the weights in the search window.

Example: To validate this bound, we consider a concrete example of a one-dimensional signal with length $n = 10^4$, corrupted by Gaussian noise with standard deviation $\sigma = 5/255$.

We apply Monte Carlo NLM to denoise a pixel using a **uniform sampling pattern** where only 5% of the pixels are sampled ($p_j = \xi = 0.05$ for all j). We set an error tolerance of $\varepsilon = 0.01$.

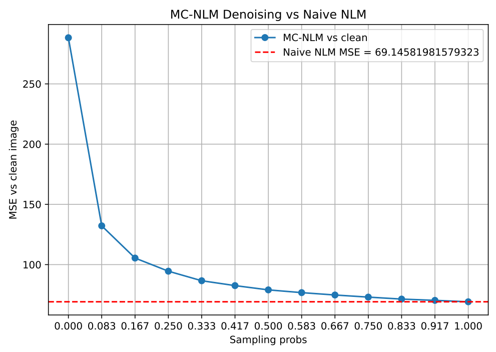
Using the derived constants:

- Average similarity weight: $\mu_B = 0.3015$
- Variance term 1: $\frac{1}{n}\sum\alpha_j^2 \approx 1.335 \times 10^{-4}$
- Variance term 2: $\frac{1}{n}\sum\beta_j^2 \approx 1.452 \times 10^{-4}$

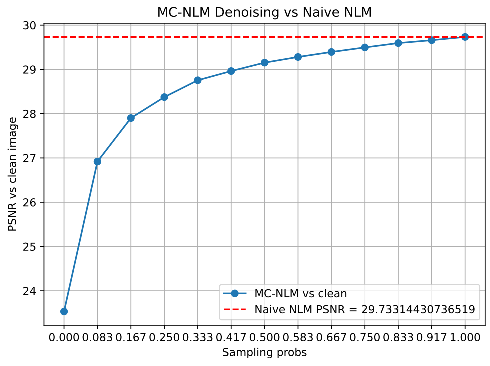
And substituting these values into the probability bound yields:

$$\mathbb{P}(|Z(\mathbf{p}) - z| > 0.01) \leq 6.516 \times 10^{-6} \tag{3.5}$$

Conclusion: This result demonstrates that even when using only **5% of the samples**, the Monte Carlo estimate stays within 1% of the true NLM result with overwhelming probability ($1 - 6.5 \times 10^{-6}$). This theoretical calculation validates the rapid MSE convergence observed in our empirical charts.



(A) MSE across different sampling probabilities



(B) PSNR across different sampling probabilities

FIGURE 4. Monte-carlo converge results

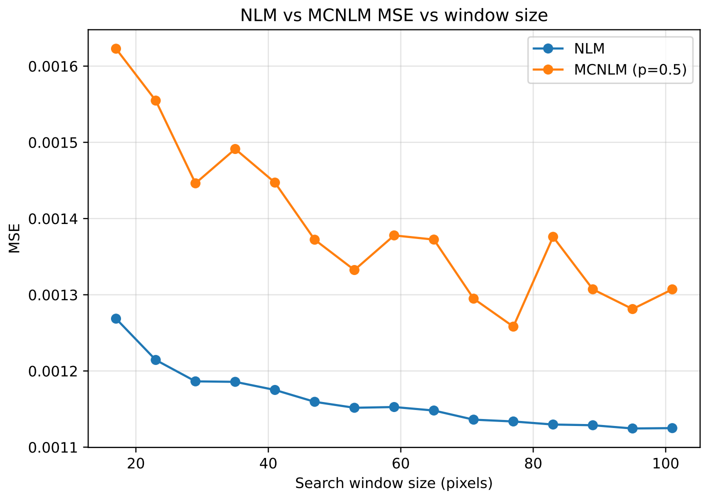


FIGURE 5. NLM vs MCNLM mse for different window sizes

4. Noise Estimation and Algorithm Implementation

In the above algorithm it is assumed that the noise standard deviation σ is known. However, in real-life applications this is not always the case. There are multiple methods for estimating this value, but we are going to focus on a simple one based on the Fast Fourier Transform. The idea is that, in the frequency domain, the high-frequency components are more likely to represent noise rather than actual image details. By analyzing these components, we can estimate the noise level. A pseudo-code of the algorithm is given below.

Algorithm 2 Gaussian Noise Standard Deviation Estimation using FFT

```
Require: Input image
1: $F \leftarrow$ FFT(image)
2: Shift the zero-frequency component $F$ to the center of the spectrum
3: Define a high-frequency mask $M$ that selects the high-frequency components
4: $H \leftarrow F \cdot M$
5: $noise \leftarrow$ Inverse FFT Shift of $H$
6: Keep only the real part of $noise$ and remove the mean.
7: return $\sigma \leftarrow std(noise)$
```

While this obviously doesn't yield perfect results like already knowing the noise level, it provides a reasonable estimate that can be used in the denoising process. We can see that the estimated noise standard deviation is close to the actual value, which is sufficient for our denoising algorithm to perform effectively. However, the blurring effect can starts to be noticeable when we compare the two images side by side.

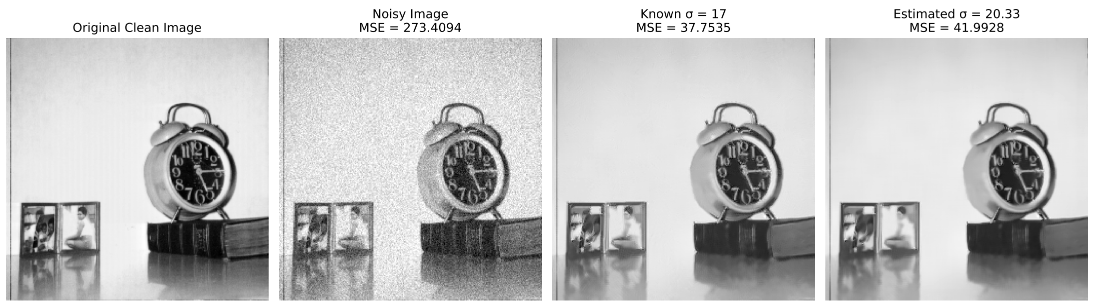


FIGURE 6. Comparison between using a known σ and an estimated one

REFERENCES

1. A. Buades, B. Coll, and J.-M. Morel, *A non-local algorithm for image denoising*, 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) (San Diego, CA, USA), vol. 2, IEEE, 2005, pp. 60–65.
2. Antoni Buades, Bartomeu Coll, and Jean-Michel Morel, *Non-Local Means Denoising*, Image Processing On Line **1** (2011), 208–212, https://doi.org/10.5201/ipol.2011.bcm_nlm.
3. Stanley H. Chan, Todd Zickler, and Yue M. Lu, *Monte carlo non-local means: Random sampling for large-scale image filtering*, IEEE Transactions on Image Processing **23** (2014), no. 8, 3711–3725.
4. Jacques Froment, *Parameter-Free Fast Pixelwise Non-Local Means Denoising*, Image Processing On Line **4** (2014), 300–326, https://doi.org/10.5201/ipol.2014.120.
5. Simon Postec, Jacques Froment, and Béatrice Vedel, *Non-local means est un algorithme de d\'ebruitage local (non-local means is a local image denoising algorithm)*, arXiv preprint arXiv:1311.3768 (2013).
6. Chunwei Tian, Menghua Zheng, Wangmeng Zuo, Bob Zhang, Yanning Zhang, and David Zhang, *Multi-stage image denoising with the wavelet transform*, Pattern Recognition **134** (2023), 109050.

7. Faiz Ullah, Kamlesh Kumar, Tariq Rahim, Jawad Khan, and Younhyun Jung, *A new hybrid image denoising algorithm using adaptive and modified decision-based filters for enhanced image quality*, Scientific Reports **15** (2025), no. 1, 8971.