ECEE6560 Final Project

Image denoising

1. Problem Description

In the field of image processing, denoising is a basic and important task. Denoising refers to the removal of noise or interference from a captured image, which may be introduced by the camera's sensor, changes in ambient lighting, or errors in the transmission process. Noise can make images look rough or randomly speckled, affecting the clarity and quality of the image.

At the same time, we also want to preserve important features in the image, such as edges, lines, and textures, during the noise removal process. These features are crucial for the human eye to recognize objects and shapes in images. Therefore, an ideal denoising process should not only reduce random noise in the image, but also ensure that the main structure and details of the image remain unchanged. In short, the goal is to clean up an image while preserving or even enhancing its important visual information.

With this in mind, the challenge is to accurately distinguish between noise and valuable details in an image in order to reduce interference while maintaining the key visual elements of the image. In practical applications, this process is important to improve the usability and quality of images, such as in medical imaging, satellite image processing, and multimedia technology.

2. Transform Mathematical Problems

Converting the problem of image denoising and edge preservation into a mathematical problem, we need to build a model to represent the image and define a mathematical expression to capture the goals we hope to achieve during image processing.

Mathematical representation of images and noise

- **Mathematical model of image**: We can think of an image as a two-dimensional function, where the brightness or color of each point is represented by the function value at that point. In mathematical terms, this means that we define an image *I* as a function $I(x, y)$ defined on a two-

dimensional plane, where *x* and *y* represent the spatial coordinates of the image, $I(x, y)$ represents the pixel intensity at coordinate (x, y)

- **Noise Model**: Noise can be viewed as an undesired random variable in an image. If $I_o(x, y)$ represents the original noise-free image, and $N(x, y)$ represents the noisy model, then the actual observed noisy image $I(x, y)$ It can be expressed as

$$I(x, y) = I_o(x, y) + N(x, y)$$

Evaluation of mathematical image quality

- **Mathematical measures of image quality**: Image quality is evaluated through different mathematical measures, such as edge preservation, detail preservation and noise suppression. Define an energy functional $E(I)$ that evaluates the quality of a given image, typically including data fidelity and image smoothness (regularization terms).
- **Data Fidelity**: Measures the consistency between the processed image and the original image. This difference can be quantified by the squared error integral

$$\iint (I(x, y) - I_o(x, y))^2 dxdy$$

  which ensuring that the processed image does not deviate too far from the original noisy image.
- **Image smoothness (regularization)**: focuses on local changes in the image, often measured by a function of the image gradient. Image gradient

$$\nabla I = (\partial I/\partial x, \partial I/\partial y)$$

  describes the spatial variation of image intensity. Denoising aims to suppress high-frequency changes caused by noise while maintaining important structural features. It can be achieved through the nonlinear function $g(||\nabla I||)$ of the gradient mode, reducing the smoothing effect in high gradient areas and increasing low gradients. Area smoothing effect.

Comprehensive energy functional

- **Construct energy functional**: Combining data fidelity and image smoothness, the energy functional can be expressed as

$$E(I) = \iint (\lambda(I(x,y) - I_o(x,y))^2 + g(||\nabla I(x,y)||))dxdy$$

- where λ is a factor that trades off data fidelity and smoothness. The function $g(||\nabla I||)$ is designed to have a selective effect on the gradient size, such as the Perona-Malik function

$$g(s) = 1/(1 + (s/K)^2)$$

where $s = ||\nabla I||$, K is a parameter that controls the edge preservation strength.

Through this mathematical processing, the image denoising problem is transformed into the problem of finding the image I that can minimize the energy functional $E(I)$, which provides a base solution for this optimization problem using partial differential equations (PDE).

3. PDE Build

In image denoising and edge preservation problems, we can use partial differential equations (PDE) to describe the changes and processing of the image. Taking total variation minimization (TV minimization) as an example, we can derive PDE from the energy functional through variational methods. The following is the derivation process:

Energy Functional and Variational Principle

Suppose our energy functional is $E(I)$, where I represents the image, and $E(I)$ is defined as:

$$E(I) = \iint (\lambda(I(x,y) - I_o(x,y))^2 + g(||\nabla I||))dxdy$$

In this expression, the first term $\lambda(I(x,y) - I_o(x,y))^2$ is the data fidelity term, and the second term $g(||\nabla I||)$ is Regularization term, used to control the smoothness and edge preservation of the image.

Export the Euler-Lagrange equation

To find the image I that minimizes the energy functional, we need to vary I and set the variational derivative equal to zero. This process leads to the formation of the Euler-Lagrange equation. For each pixel position (x, y), the variational derivative gives the Euler-Lagrange equation of the form:

$$\delta E/\delta I = -2\lambda(I - I_o) + \nabla \cdot (g'(||\nabla I||)\nabla I||\nabla I||) = 0$$

Here, $\nabla\cdot$ represents divergence, which represents the degree to which the vector field (in this case, the image gradient ∇I) diverges. The function $g'(||\nabla I||)$ is the derivative of $g(||\nabla I||)$ with respect to $||\nabla I||$.

Understand the physical meaning of equations

This Euler-Lagrange equation combines the requirements of data fidelity and image smoothness. The data fidelity term $-2\lambda(I - I_o)$ causes the denoised image I to be close to the original noisy image I_o. The regularization term $\nabla \cdot (g'(\|\nabla I\|)\nabla I/\|\nabla I\|)$ pushes the image to achieve a smooth effect while maintaining edges.

When choosing an appropriate function in $g(||\nabla I||)$(such as the nonlinear function in the Perona-Malik model), the regularization term helps to reduce smoothing at the edges, thereby preserving important image features while Add smoothing to edge areas to remove noise.

Implementation details

In numerical implementation, this PDE usually needs to be discretized. The spatial derivative ∇I and the divergence $\nabla\cdot$ need to be approximated by numerical methods such as finite difference methods. Choosing an appropriate differencing scheme and step size is crucial to ensure the stability and accuracy of the numerical solution.

Through the above detailed discussion, we have demonstrated the conversion process from a mathematical model to an actual solvable PDE, and gained an in-depth understanding of the physical meaning and mathematical principles in this process. This PDE captures the key elements of image denoising and provides us with a powerful tool to solve the problem mathematically.

4. Discretization and computer implementation

To implement a partial differential equation (PDE) on a computer, we need to discretize it. Discretization is the process of converting a continuous model into

a discrete numerical approximation, making it solvable on a computer.

Space discretization

- **Choose a difference scheme**: When discretizing a PDE in space, we need to use a difference scheme to approximate the derivatives. The central difference scheme was chosen in my project because it provides second-order accuracy and considers the left and right information of the point relatively balanced. The central difference scheme approximates the derivative using the difference between neighboring points:

$$\frac{\partial I}{\partial x} \approx \frac{I(x+1,y) - I(x-1,y)}{2\Delta x}$$

  where Δx is the grid spacing. This scheme can better capture the local changes in the image and is suitable for edge-preserving denoising.

  - **Accuracy**: Central difference provides higher accuracy than forward difference or backward difference. The error term is $O(\Delta x^2)$, where Δx is the space of the grid step length.
  - **Symmetry**: Central difference utilizes the information on both sides of the point, which is particularly suitable for data types such as images, because it can process the surrounding information of the pixel more evenly and avoid introducing directional deviations.

- **Handling Boundary Conditions**: At the edges of the image, we may need special processing because not all points have left and right neighbors. Typically, this problem can be solved using pixel extrapolation, mirroring, or periodic boundary conditions.

Time discretization

- **Explicit method**: The discretization in time can choose an explicit method, such as the forward Euler method, where the new image I^{n+1} is given by the current image I^n Add a time step Δt multiplied by the derivative to update:

$$I^{n+1}(x,y) = I^n(x,y) + \Delta t \cdot \left(-\nabla \cdot \left(\frac{\nabla I}{|\nabla I|} \right) \right)$$

The explicit method is simple and easy to implement, but care needs to be taken to select an appropriate time step Δt to ensure numerical stability.

- **Selection of time step**: The choice of time step Δt is crucial to the stability and efficiency of the algorithm. Too large Δt may cause the calculation to be unstable, while too small Δt may cause the calculation process to be slow. It is usually necessary to select the time step according to the Courant–Friedrichs–Lewy (CFL) condition to ensure that the calculation at each step is stable. I chose a larger step size for my project because of two main considerations:
    - **Larger time step**: Choosing a larger time step (e.g. *timestep*=5) can speed up reaching a steady state in numerical simulations, which is beneficial for fast denoising.
    - **Stability considerations**: Although a large time step can improve calculation efficiency, it must be selected while ensuring the stability of the algorithm. Through the application of semi-implicit and AOS algorithms, your code enhances the stability of the numerical solution, allowing the use of larger time steps.

Application of AOS algorithm

The Alternating Direction Implicit (AOS) algorithm plays a key role in your implementation. The advantages of this algorithm are:

- **Computational efficiency**: By decomposing a multi-dimensional problem into a set of one-dimensional sub-problems, the AOS algorithm can significantly reduce the computational complexity. Solving independently in each direction makes the problem easier to process in parallel, improving efficiency.
- **Numerical Stability**: The AOS algorithm improves the stability of numerical methods by implicitly processing some terms. This is particularly important when dealing with PDEs involving time evolution, as it prevents numerical instabilities caused by improperly chosen time steps.

Implementation details

In my code implementation, iterative processing in two directions (horizontal and vertical) is used, and the Thomas algorithm (catch-up method) is used in each direction to solve the system of linear equations. This method effectively

combines the advantages of AOS and the accuracy of central difference, providing a solid numerical foundation for denoising algorithms.

Through the above method, I achieved efficient and stable image denoising processing. This implementation strategy ensures that the accuracy and stability of the algorithm can be maintained while achieving efficient calculations.

5. Experiments and Analysis

5.1 Experimental Design Part

Purpose

The main goal of this experiment is to evaluate and compare the performance of denoising algorithms under different noise types (Gaussian noise and salt-and-pepper noise) and different parameter settings. By adjusting the number of iterations and step size, this experiment aims to observe how the denoising effect changes as parameters change, and to quantitatively evaluate the denoising results using PSNR and SSIM indicators.

Experimental Materials

- **Image Selection**: This experiment selected three test images to examine the performance of the denoising algorithm on different image contents.
- **Noise Type**:
    - **Gaussian Noise**: Taking 0 as the mean and standard deviation ranging from 10 to 100, evaluate the impact of different noise intensities on the denoising effect.
    - **Salt and Pepper Noise**: The noise density changes from 0.1 to 0.5 to explore the effectiveness of denoising algorithms with different densities.

Gaussian Noise

Image 1: Lady

Original Image

Gaussian Noise Image

Image 2: Woman

Original Image

Gaussian Noise Image

Image 3: Stephen Curry

Original Image

Gaussian Noise Image

Salt & Pepper Noise

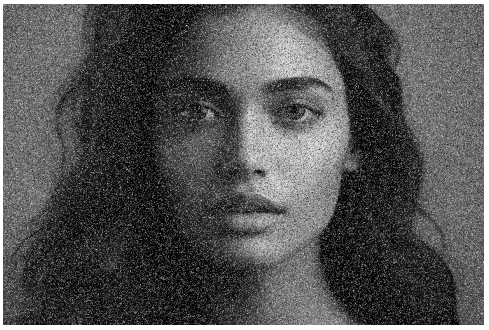
Image 1: Lady

Original Image

Salt & Pepper Noise Image

Image 2: Woman

Original Image



Salt & Pepper Noise Image

Image 3: Stephen Curry

Original Image

Salt & Pepper Noise Image

Experimental method

- **Noise Addition**: Use the OpenCV library to add Gaussian noise and salt-and-pepper noise to the original image respectively in the Python environment.
- **Denoising algorithm**: Adopts a regularization method based on partial differential equations, specifically implemented using a semi-implicit scheme combined with the AOS algorithm. This algorithm can effectively balance image detail preservation and noise suppression.
- **Parameter settings**:
  - **Number of iterations**: from 10 to 100 times to observe the changes in denoising effect as the number of iterations increases.
  - **Step Size**: Ranging from 1 to 10, analyze how the step size affects the convergence speed and quality of the denoising effect.

Experiment process

1. **Noise image generation**: Generate different levels of Gaussian noise and salt-and-pepper noise images for each test image.
2. **Denoising processing**: Apply the denoising algorithm to each noisy image, and record the denoised images under different iteration times and step size settings.

3. **Performance Evaluation**: Calculate the PSNR and SSIM indicators between each denoised image and the original image to evaluate the denoising quality.
4. **Data Recording and Analysis**: Perform statistics and analysis on the performance indicators of all denoising results, and draw the change curves of energy evolution, PSNR and SSIM to visually display the trend of denoising effect with parameter changes.

Through the above design, the experiment can not only systematically evaluate the performance of the denoising algorithm under different conditions, but also provide an in-depth understanding of the specific impact of parameter selection on the denoising effect, thus providing a scientific basis for parameter optimization in practical applications.

5.2 Result Display

This section shows the results of denoising experiments, including denoised images, energy evolution, and changes in PSNR and SSIM indicators. The experiments involved adding two types of noise (Gaussian noise and salt and pepper noise) to three different test images. The following is the specific result display and analysis:

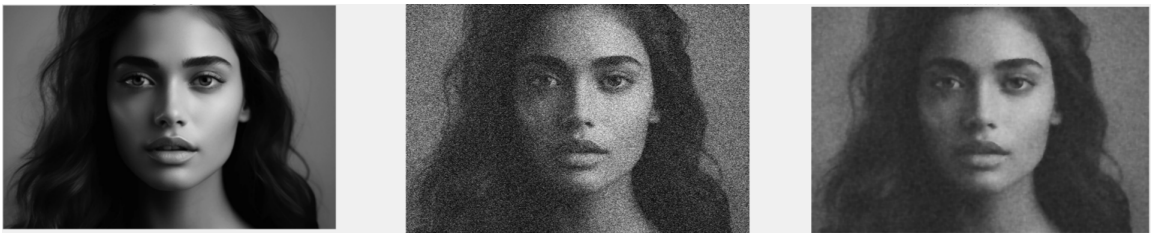
1. Display of denoising image results

This section will use comparison charts to demonstrate the denoising effects under different noise types and different parameter settings. The following is a detailed presentation:

Gaussian noise denoising results:

- Image 1 shows the original image, the image after adding Gaussian noise, and the denoised image after 7 iterations. Similarly, images two and three also show the same process, reflecting the effect of the denoising algorithm when dealing with Gaussian noise.

Original Image | Noisy Image | Denoised Image



Original Image | Noisy Image | Denoised Image

Original Image | Noisy Image | Denoised Image

Salt and pepper noise denoising results:

- For salt and pepper noise, Image 1, Image 2 and Image 3 also show the denoising results from the original state to noise addition to 7 iterations. These images reveal the denoising performance of salt and pepper noise at different numbers of iterations.

Original Image | Noisy Image | Denoised Image

Original Image | Noisy Image | Denoised Image

Original Image | Noisy Image | Denoised Image

Denoising effect for different iterations

In this part we use Lady image as a example for showing how the number of iterations will affect denoising.

Gaussian Noise

Noisy Image | Iteration 1 | Iteration 10 | Iteration 40

Salt & Pepper Noise

Noisy Image | Iteration 1 | Iteration 10 | Iteration 40

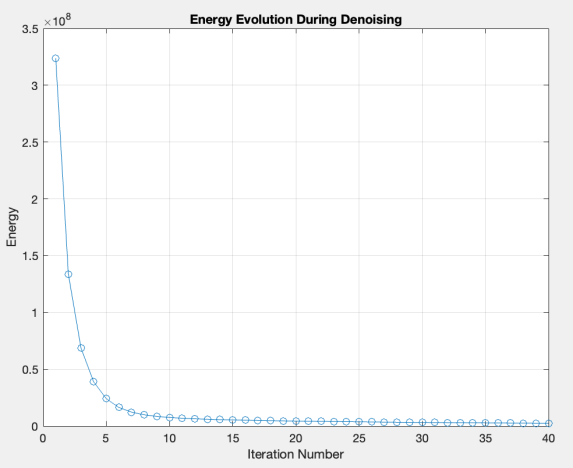
From the denoised image we can see that we increase the number of iterations, the denoising effect of the image does not get better, but becomes blurrier. This shows that the number of iterations has a saturation value, and too many iterations will have side effects.

2. Display of performance evaluation indicators

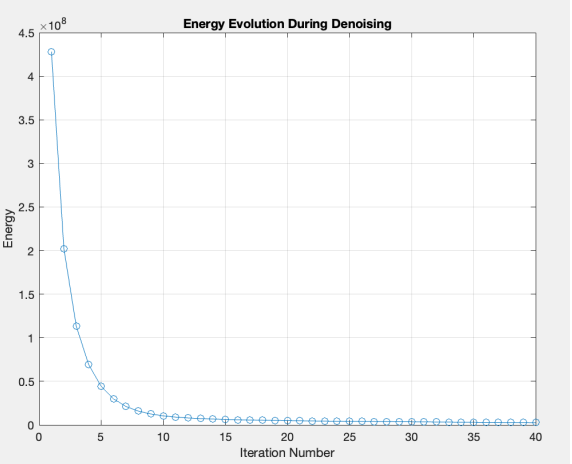
For each denoising experiment, PSNR and SSIM metrics are used to quantify the denoising effect. Here's how the indicators are displayed in detail:

Energy Evolution:

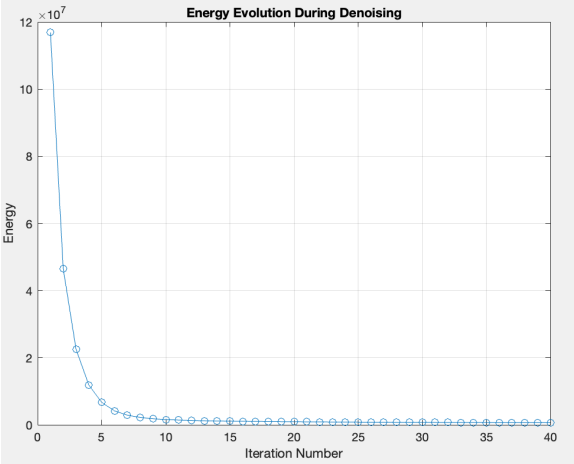
- The energy evolution curve shows the energy changes of the three test images during the denoising process from the first iteration to the last iteration. The image shows that the energy value gradually decreases with the number of iterations, showing the increase in image smoothness during denoising.



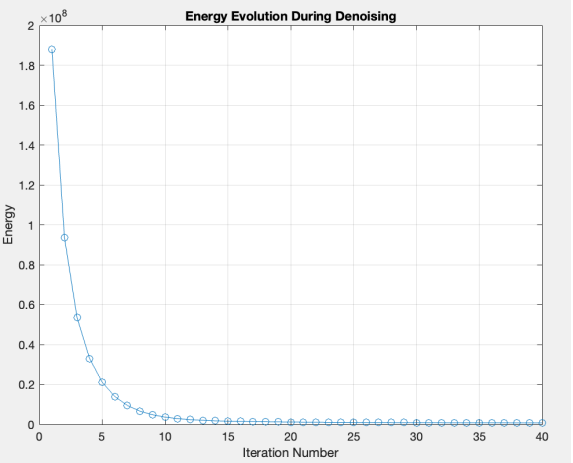
Lady Image Gaussian Noisy



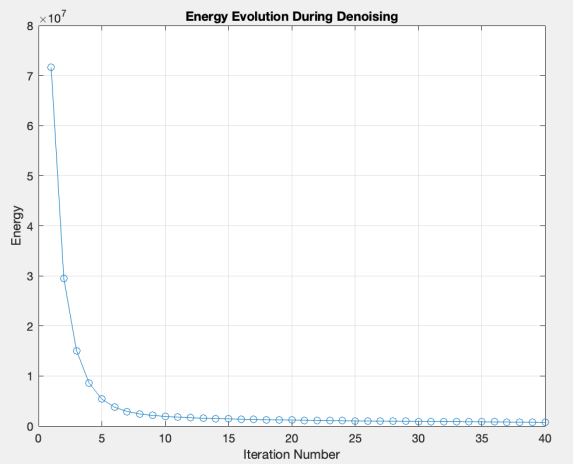
Lady Image Salt & Pepper Noisy



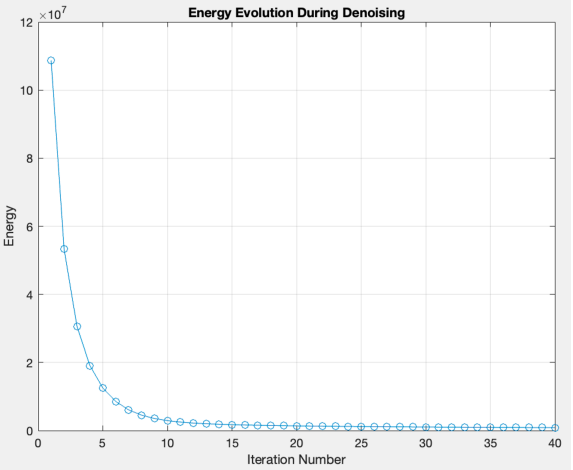
Woman Image Gaussian Noisy



Woman Image Salt & Pepper Noisy



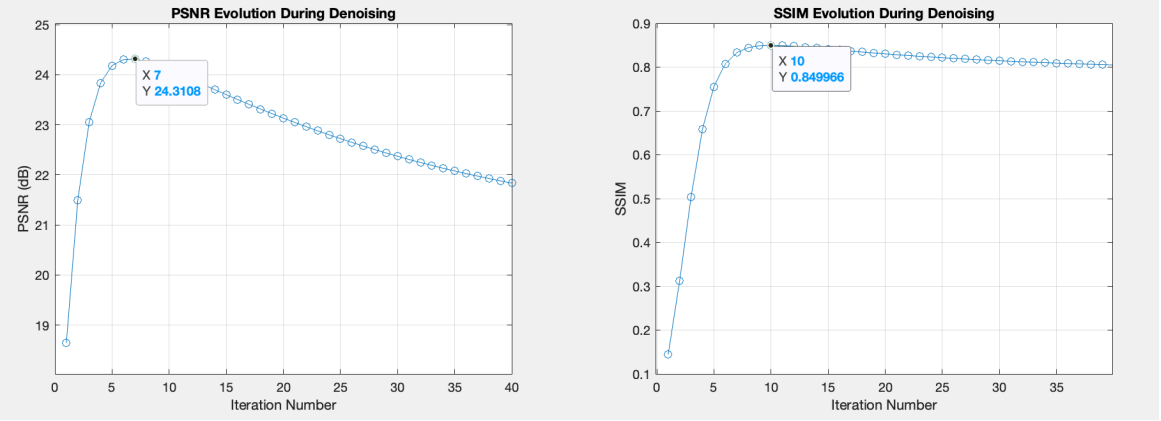
Curry Image Gaussian Noisy



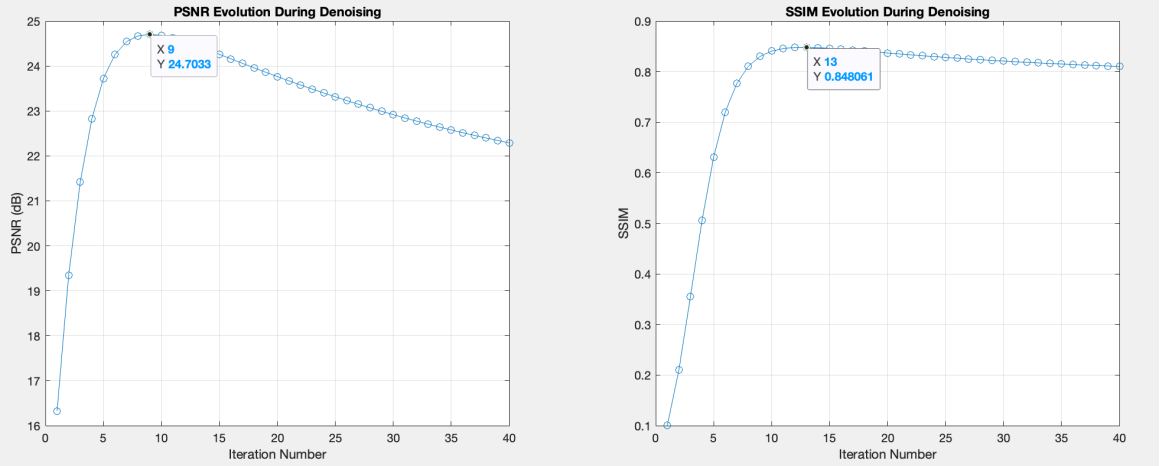
Curry Image Salt & Pepper Noisy

PSNR and SSIM evolution:

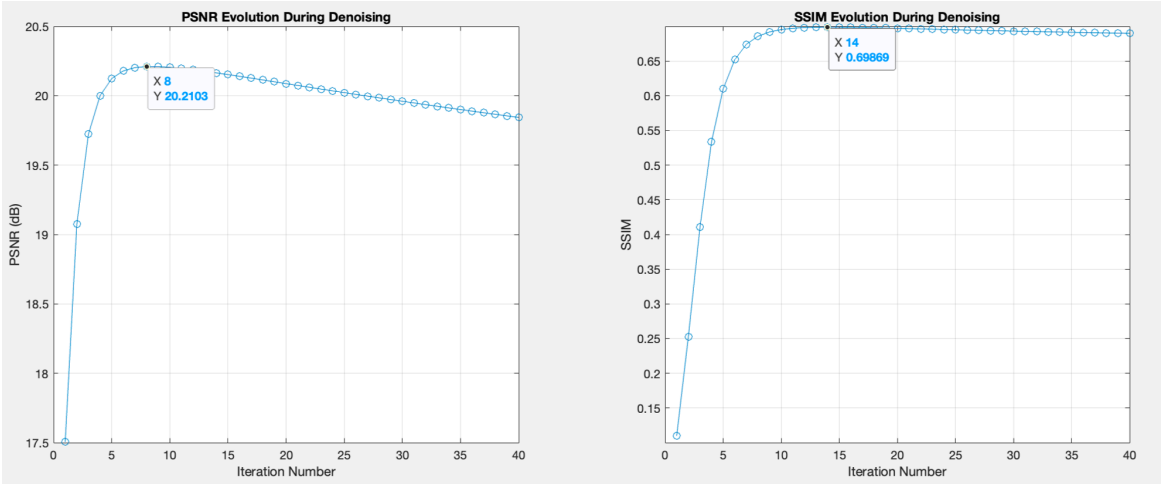
- The PSNR evolution curve (image five) and SSIM evolution curve (image six) show the changes in these two indicators during the denoising process. All three test images reached the optimal value of PSNR after 7 iterations and the optimal value of SSIM after 9 iterations. These two indicators begin to show a downward trend in subsequent iterations, indicating that excessive iteration may cause over-smoothing, thereby reducing the visual quality of the image.



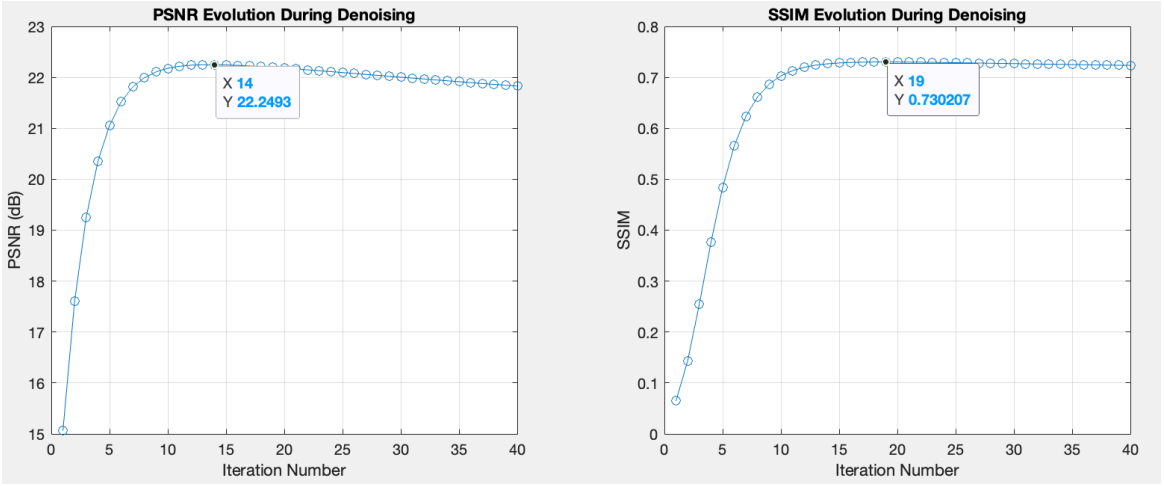
Lady Image Gaussian Noisy



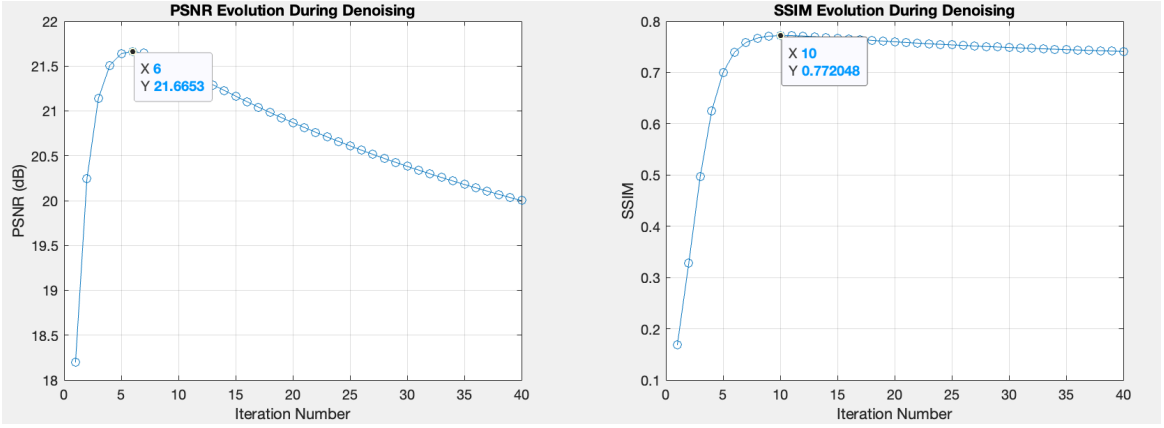
Lady Image Salt & Pepper Noisy



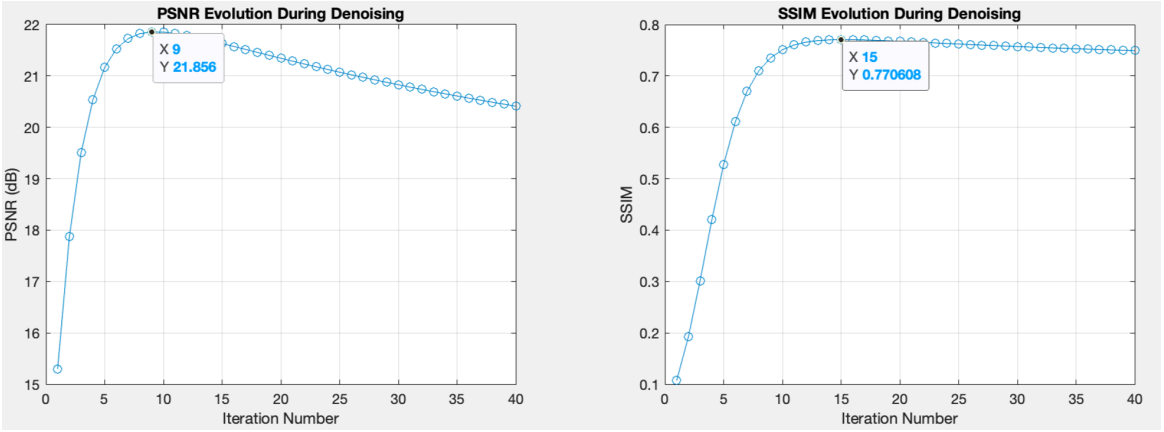
Woman Image Gaussian Noisy



Woman Image Salt & Pepper Noisy



Curry Image Gaussian Noisy



Curry Image Salt & Pepper Noisy

3. Parameter impact analysis

In this experiment (take lady image as an example), the number of iterations and step size are key parameters that affect the denoising effect. By adjusting these parameters, we observe that PSNR reaches the optimum after 7 iterations, while SSIM reaches its peak after 9 iterations. This finding reminds us that in practical applications, the appropriate number of iterations and step size should be selected based on specific denoising requirements and image characteristics to avoid image quality degradation caused by over-processing.

Through the above detailed result display and analysis, this report demonstrates the performance of the denoising algorithm under different test images and noise conditions, providing a valuable reference for future algorithm optimization and parameter adjustment.

6. Experience and Discussion

Lessons Learned

Advantage

1. **Strong edge preservation performance**:
    - By using the gradient-based nonlinear edge-preserving function $g(||\nabla I||)$, experiments show that our PDE method effectively preserves image edges while denoising. This demonstrates the superiority of nonlinear diffusion methods in processing images with high contrast edges.

2. **Stability and efficiency of algorithm**:
    - Using the Alternating Direction Implicit (AOS) method and the semi-implicit time stepping strategy greatly improves the stability of the numerical solution and allows us to use larger time steps in the experiment, which improves the efficiency of the algorithm while maintaining A good denoising effect.

Limitations

1. **Sensitivity to Noise Types**:
    - It was found in experiments that although the model performs well for certain types of noise (such as Gaussian noise), it does not perform well for other types of noise such as salt and pepper noise. This exposes the model's dependence on noise types and the limitations of nonlinear diffusion functions in dealing with multiple noise types.
2. **Parameter sensitivity**:
    - Parameters in PDE implementation (such as diffusion coefficient k) have a significant impact on the final denoising effect. Improper parameter selection can result in over-smoothing or retaining too much noise. This shows that the model is more sensitive to parameters and requires finer adjustments to adapt to different image and noise conditions.

Future improvement directions

1. **Processing of multiple types of noise**:
    - In order to improve the model's adaptability to different types of noise, you can consider introducing more types of diffusion functions or combining other denoising techniques (such as Bolt domain filtering or deep learning methods) to better handle different types of noise.
2. **Adaptive parameter selection**:
    - Develop an adaptive mechanism to dynamically adjust parameters in PDE, such as automatically adjusting the diffusion coefficient k according to the local characteristics of the image. This can be achieved through machine learning techniques, such as using training data to predict optimal parameters.
3. **Advanced discretization technology**:

- Consider using higher order numerical discretization methods to improve solution accuracy. For example, the introduction of finite element methods or multi-grid techniques may provide better convergence performance and higher numerical stability.

With these improvements, we expect to develop more powerful and versatile image denoising tools in the future while reducing sensitivity to parameter selection and improving robustness to various noise conditions.