A New Anisotropic Fourth-Order Diffusion Equation Model Based on Image Feature for Image Denoising

Ying Wen* Jiebao Sun † Zhichang Guo‡

Abstract

Image denoising has always been a challenging task. For that, one of the most effective methods is variational PDE. Inspired by the LLT model, we first propose a new adaptive LLT model by adding a weighted function and then propose a class of fourth-order diffusion equations based on the new functional. Because of the adaptive function, the new functional is better than the LLT model and other fourth-order models in edge preserving. Generalizing the Euler-Lagrange equation of the new functional, we discuss a new fourth-order diffusion framework for image denoising. Different from the other fourth-order diffusion models, the new diffusion coefficients depend on the first-order and second-order derivatives, which can preserve edges and smooth image, respectively. In the numerical aspects, we first design an explicit scheme for the proposed model. However, fourth-order diffusion equations require more strict stability conditions and the number of iteration is considerable. Consequently, we apply the fast explicit diffusion (FED) to our proposed explicit scheme for reducing the time consumption. Furthermore, a semi-implicit scheme for the new equation is discussed and is accelerated by the additional splitting (AOS) algorithm. The AOS algorithm is most efficient in our all algorithms. Finally, compared with other models, the new model shows the effectiveness and efficiency.

Keywords: image denoising; adaptive functional; fourth-order diffusion equation

1 Introduction

Image denoising has always been an essential and challenging task in the field of image processing because the result of denoising renders the later phases. The noise can be considered as unpredictable and subject to a random error of a specific probability distribution, which can be described by the probability distribution function and probability density function. The additive noise is independent of the input signal.

*School of Mathematics, Harbin Institute of Technology, Harbin, China. wenyinghitmath@gmail.com

†School of Mathematics, Harbin Institute of Technology, Harbin, China. sunjiebao@hit.edu.cn

‡School of Mathematics, Harbin Institute of Technology, Harbin, China. mathgzc@gmail.com

Giving a noisy image $f : \Omega \to \mathbb{R}$, which is defined on the domain $\Omega \subset \mathbb{R}^d$, the additive noise model is

$$f = u + \eta,$$

where u is the true image and η is the additive noise. Image denoising is used to recover the true image u from an observed noisy image f.

During the last three decades, a variety of methods have been developed to deal with the above inverse problem. Among these, variational method [1, 2, 3, 4, 5] and partial differential equation (PDE) method [6, 7, 8] are most extensively used techniques. In 1992, Rudin, Osher, and Fatemi [1] propose a nonlinear total variation model for noise removal, which indicates this issue is equivalent to the following minimisation problem (TV),

$$\min_u E(u) = \int_\Omega |\nabla u| + \lambda \int_\Omega (u - f)^2 , \tag{1.1}$$

where $\lambda > 0$ is a tuning parameter. The first term, the total variation of u, is a regularizer. The second term is a fidelity term which ensures that denoised image u will be close to the original image f. This total variational regularizer helps to prevent the noise from staying in the denoised image u because the noise part yields a sizeable total variation of u. At the same time, this regularizer also allows discontinuities of u.

In 2000, Chen, Marquina, and Mulet [9] describe the phenomenon that TV model preserves edges well, but the images resulting from the application of this technique in the presence of noise are often piecewise constant, thus the more delicate details in the original image may not be recovered satisfactorily, and ramps (affine regions) will give stairs (piecewise constant regions). Therefore they propose an improved model that adds a nonlinear fourth-order diffusive term to the Euler-Lagrange equations of the TV model.

Some properties of the fourth-order PDEs show its unique advantages in image processing by theoretical analysis in [10, 11]. Fourth-order PDE model can efficiently overcome the staircase effects. As in [12], one can prove that the solution of fourth-order PDE model converges to a piecewise harmonic image in a specific condition. There are many denoising methods based on fourth-order PDE. In 2003, Lysaker, Lundervold, and Tai [2] propose a fourth-order noise removal model (LLT),

$$\int_\Omega |u_{xx}| + |u_{yy}| \, \mathrm{d}x\mathrm{d}y + \lambda \int_\Omega (u - f)^2 \tag{1.2}$$

or

$$\int_\Omega \sqrt{|u_{xx}|^2 + |u_{xy}|^2 + |u_{yx}|^2 + |u_{yy}|^2} \mathrm{d}x\mathrm{d}y + \lambda \int_\Omega (u - f)^2 . \tag{1.3}$$

LLT model tries to minimize the total variational norm of ∇u instead of u [2]. The parabolic equation obtaioned by the LLT model is

$$\frac{\partial u}{\partial t} = -\left(\frac{u_{xx}}{|u_{xx}|}\right)_{xx} - \left(\frac{u_{yy}}{|u_{yy}|}\right)_{yy} - \lambda (u - u_0) , \tag{1.4}$$

or

$$\frac{\partial u}{\partial t} = -\left(\frac{u_{xx}}{|D^2 u|}\right)_{xx} - \left(\frac{u_{xy}}{|D^2 u|}\right)_{xy} - \left(\frac{u_{yx}}{|D^2 u|}\right)_{yx} - \left(\frac{u_{yy}}{|D^2 u|}\right)_{yy} - \lambda (u - u_0) , \tag{1.5}$$

where $|D^2u| = \left(|u_{xx}|^2 + |u_{yx}|^2 + |u_{xy}|^2 + |u_{yy}|^2\right)^{1/2}$. The LLT model handles smooth signals better than the TV model (1.1), and has no staircase effect. However, it seems to be hard for the LLT model to maintain discontinuities in the jump areas of the image.

There are much research pursuing to dealing with the problem that fourth-order equation models could not preserve edges well. For that purpose, Lysaker and Tai [13] uses a weighting function in an iterative way to combine the solutions of the LLT model and the TV model in 2006. However, the weighting function in [13] is not easy to construct. Base on that, Li, Shen, Fan, and Shen [14] construct an energy functional combining the LLT model and TV model by two weighted functions in 2007. There are also some adaptive fourth-order models [15] introducing a diffusivity function to adaptively manipulates the amount of smoothing. Most of these models focus on the LLT model (1.3) rather than (1.2). However, because (1.3) has cross-derivative terms, (1.3) is more complicated than (1.2), which causes more strict CFL condition and extra computation cost. Another classical fourth-order model is [16] proposed by You and Kaveh in 2000, which approximates the noisy image with a piecewise planar image by seeking to minimize a functional which is is an increasing function of the absolute value of Laplacian of the image and deduces an anisotropic fourth-order diffusion equation. But the model of [16] would leave speckle pixels, which cause bad vision [17]. Inspired by [16], many researchers try to design more rational anisotropic fourth-order equations [8, 18]. At the same time, some researchers further employ the fourth-order diffusion term to other image processing tasks, image inpainting [19, 20, 21, 22], image restoration [23, 24], image zooming [25] and image reconstruction [26, 27]. Although fourth-order equation based image denoising models has been thoroughly investigated for many years, technical challenges remain.

In this paper, inspired by the LLT model, we propose a new fourth-order diffusion equation for image denoising. First, using the gradient module of the image to design a weighted function, we recommend an adaptive LLT model. The weighted function indicated where is the edge in the image, thus the new model can preserve edge in this region. The developing equations of the proposed adaptive LLT model are deduced, and a 1D experiment shows the essential difference between the LLT model and our adaptive LLT model in edge preserving. Second, we generalize the model to a class of fourth-order diffusion equations. A new fourth-order diffusion framework is proposed for image denoising, and the diffusion coefficient is designed by the first-order and second-order derivatives of the image to make full use of image feature, which the first-order derivative could better detect edges with anti-performance, and the part of second-order derivative is the core to overcome staircase effect. At last, we design some algorithms for numerical implementation. We propose a finite difference explicit scheme and semi-implicit scheme for the proposed fourth-order diffusion equations. And then using the idea of the fast explicit scheme (FED) and additional operator splitting (AOS) to accelerate these schemes. Some of them are proved to be very efficient.

The paper is organised as followed. In section 2, the proposed adaptive LLT model is presented. Section 3 gives the details of our new fourth-order PDE noise removal model. A series of numerical methods of our novel model are discussed in section 4. Finally, section 5 is devoted to some experimental results for demonstrating the new model.

2 The New Proposed Functional

In this section, we firstly propose a new adaptive LLT model. For better preserve structures of the image, the new model does different processing in edges and homogeneous areas. We also prove that the result of new adaptive LLT model is piecewise planar, and it is the reason why our model could preserve structures without staircase effect like the TV model. Secondly, we deduce the developing equations of the proposed functional. Moreover, 1D experiments confirm that new fourth-order diffusion equation is essentially different from the LLT model in structure preservation, edges and flat areas preservation, compared with other fourth-order PDE models.

2.1 The New Adaptive LLT Model

Inspired by the LLT model, we propose the following adaptive LLT model

$$E(u) = \int_{\Omega} \alpha(f)\left(|u_{xx}| + |u_{yy}|\right) \mathrm{d}x\mathrm{d}y + \lambda \int_{\Omega} (u-f)^2 \, \mathrm{d}x\mathrm{d}y, \tag{2.1}$$

where $\alpha(f)$ is a feature detection function,

$$\alpha(f) = \frac{1}{1 + \left(|\nabla G_\sigma * f| / K\right)^2}, \tag{2.2}$$

$G_\sigma(\cdot) = \frac{1}{2\pi\sigma^2}\exp(-\frac{|\cdot|^2}{2\sigma^2})$ denotes Gaussian function with standard deviation σ, and K is a parameter to adjust the contrast of features. In addition, the equation (2.2) is usually used as a diffusion coefficient in diffusion equations [7]. In our model, we only describe the model for 2-D problems. The generalization to d-D can be given by

$$E(u) = \int_{\Omega} \alpha(f) \left(\sum_{i=1}^{d} |u_{x_i x_i}| \right) \mathrm{d}x + \lambda \int_{\Omega} (u-f)^2 \, \mathrm{d}x.$$

From equation (2.2), notice that the function $\alpha(f)$ is limited in $[0,1]$. For given K, $\alpha(f)$ tends to 0 at the points where $|\nabla G_\sigma * f|$ is very large. It represents edge regions in the image, and we denote it as Γ. $\alpha(f)$ approaches to 1 at the points where $|\nabla G_\sigma * f|$ tends to 0. It represents homogeneous regions, and we denote it as Ω_h.

Consider the following functional,

$$\widetilde{E}(u) = E_1(u) + E_2(u), \tag{2.3}$$

where

$$E_1(u) = \int_{\Omega_h} \left(|u_{xx}| + |u_{yy}|\right) \mathrm{d}x\mathrm{d}y + \lambda \int_{\Omega_h} (u-f)^2 \, \mathrm{d}x\mathrm{d}y,$$

and

$$E_2(u) = \lambda \int_{\Gamma} (u-f)^2 \, \mathrm{d}x.$$

In ideal case, $\alpha(f) = 0$, as $x \in \Gamma$, and $\alpha(f) = 1$, as $x \in \Omega_h$. So the functional (2.3) can be expressed as

$$\widetilde{E}(u) = \int_{\Omega} \alpha(f)\left(|u_{xx}| + |u_{yy}|\right) \mathrm{d}x\mathrm{d}y + \lambda \int_{\Omega} (u-f)^2 \, \mathrm{d}x. \tag{2.4}$$

which imply that $\widetilde{E}(u) = E(u)$. Therefore, the functional (2.1) and (2.3) is equivalent. Since E_1 and E_2 are two independent sub-problems, minimizing $\widetilde{E}(u)$ is equivalent to minimizing $E_1(u)$ and $E_2(u)$, respectively. Minimizing the functional $E_2(u)$ means that u is the original image f in the points $(x, y) \in \Gamma$, so the edges can be preserved. At the same time, E_1 is the LLT model (1.2) restricted to Ω_h, and the LLT model is famous for its ability that removes noise well without staircase effects in flat areas. As a result, our new model could smooth noise in homogeneous regions Ω_h like the LLT model and preserve jumps in edge regions Γ.

Similar to [12] and [16], we give the following theorem.

Theorem 1. *Suppose that u is piecewise smooth. Then u is piecewise planar if and only if*

$$\int_{\Omega} \alpha(f) \left(|u_{xx}| + |u_{yy}|\right) dx\, dy = 0. \tag{2.5}$$

Proof. ($\Rightarrow$)Let u be piecewise planar as following

$$u(x, y) = \sum_{i=1}^{n} u_i(x, y), \tag{2.6}$$

where $\Omega_i, i = 1, 2, 3, \cdots, n$ is a partition of Ω and

$$u_i(x, y) = \begin{cases} \text{planar}, & (x, y) \in \Omega_i, \\ 0, & \text{otherwise.} \end{cases}$$

Let us denote the interior and the boundary of Ω_i by Ω_i^o and $\partial\Omega_i$, respectively. Define $\Gamma_{ij} = \partial\Omega_i \cap \partial\Omega_j$, the edge between neighboring subregions Ω_i and Ω_j. Then

$$|u_{xx}| = |u_{yy}| = 0, \quad \forall (x, y) \in \Omega_i^o, \quad i = 1, 2, \cdots, n. \tag{2.7}$$

and the range of $\alpha(f)$ is $[0, 1]$. On the other hand, ∇u_i are constant vectors on each $\Omega_i, i = 1, \cdots, n$, and

$$\nabla u_i \neq \nabla u_j, \quad \forall (x, y) \in \Gamma_{i,j}, \quad i, j = 1, \cdots, n, \tag{2.8}$$

So we have among of $|u_{xx}(x, y)|$, $|u_{yy}(x, y)|$ is infinite, $\forall (x, y) \in \cup_{ij}\Gamma_{ij}$. At the same time, $\cup_{ij}\Gamma_{ij}$ represents edge regions, and $\alpha(f) = 0$ in edge regions. Therefore $\alpha(f)\left(|u_{xx}| + |u_{yy}|\right)$ is bounded (0 or finite value), $\forall (x, y) \in \cup_{ij}\Gamma_{ij}$, which indicates that the vaule of $|u_{xx}|$ and $|u_{yy}|$ is not constrained in edge regions. Note that $\cup_{ij}\Gamma_{ij}$ is a set of measure zero. Hence the formula (2.5) is established.

($\Leftarrow$)Let u be piecewise smooth and satisfy (2.5). Then $|u_{xx}|$, $|u_{yy}|$ must be zero at least at all interior points of the subregions in which u is smooth, which implies that u is piecewise planar. This completes the proof. □

The above Theorem 1 proves that the piecewise planar image is the only minimizer of new proposed functional (2.1) without fidelity term, which theoretically shows adaptive functional (2.1) could preserve structures. However, the minimizer of the LLT model without fidelity term is the only globally planar from [12], which indicates the LLT model would convert jumps to slops. These illustrate that our model is essentially different from the LLT model.

2.2 The Fourth-Order Diffusion Equation Based on Functional (2.1)

The Euler-Lagrange equation of the functional (2.1) is as follows,

$$\left(\alpha(f)\frac{u_{xx}}{|u_{xx}|}\right)_{xx}+\left(\alpha(f)\frac{u_{yy}}{|u_{yy}|}\right)_{yy}+\lambda\left(u-f\right)=0,\ \text{ in }\ \Omega,$$

and the boundary conditions are given as

$$\begin{aligned}&\frac{\partial u}{\partial\overrightarrow{n}}=0,\ \text{ on }\ \partial\Omega\\&\left(\alpha(f)\frac{u_{xx}}{|u_{xx}|}\right)_{x}n_1+\left(\alpha(f)\frac{u_{yy}}{|u_{yy}|}\right)_{y}n_2=0,\ \text{ on }\ \partial\Omega\end{aligned}\tag{2.9}$$

where $\overrightarrow{n}=(n_1,n_2)$ is the outward normal direction on $\partial\Omega$. To solve Euler-Lagrange equation, we consider the gradient descent flow,

$$\begin{aligned}&\frac{\partial u}{\partial t}=-\left(\alpha(f)\frac{u_{xx}}{|u_{xx}|}\right)_{xx}-\left(\alpha(f)\frac{u_{yy}}{|u_{yy}|}\right)_{yy}-\lambda\left(u-f\right),\ \text{ in }\ \Omega\times(0,T)\\&\frac{\partial u}{\partial\overrightarrow{n}}=0,\ \text{ on }\ \partial\Omega\times(0,T)\\&\left(\alpha(f)\frac{u_{xx}}{|u_{xx}|}\right)_{x}n_1+\left(\alpha(f)\frac{u_{yy}}{|u_{yy}|}\right)_{y}n_2=0,\ \text{ on }\ \partial\Omega\times(0,T)\\&u\left(x,y,0\right)=f(x,y),\ \text{ on }\ \Omega.\end{aligned}\tag{2.10}$$

Next, we do some experiments for better verifing the essential difference between our new functional (2.1) and the LLT model. In fairness, we condiser 1D case and set $\lambda=0$ in both the LLT model (1.2) and the new adaptive LLT model (2.1). The developing equations of both without fidelity term are given as follows, respectively,

$$\begin{aligned}&\frac{\partial u}{\partial t}=-\left(\frac{u_{xx}}{|u_{xx}|}\right)_{xx},\ \text{ in }\ \Omega\times(0,T)\\&\frac{\partial u}{\partial n}=0,\ \text{ on }\ \partial\Omega\times(0,T)\\&\left(\frac{u_{xx}}{|u_{xx}|}\right)_{x}n=0,\ \text{ on }\ \partial\Omega\times(0,T)\\&u\left(x,0\right)=f(x),\ \text{ on }\ \Omega,\end{aligned}\tag{2.11}$$

and

$$\begin{aligned}&\frac{\partial u}{\partial t}=-\left(\alpha(f)\frac{u_{xx}}{|u_{xx}|}\right)_{xx},\ \text{ in }\ \Omega\times(0,T)\\&\frac{\partial u}{\partial n}=0,\ \text{ on }\ \partial\Omega\times(0,T)\\&\left(\alpha(f)\frac{u_{xx}}{|u_{xx}|}\right)_{x}n=0,\ \text{ on }\ \partial\Omega\times(0,T)\\&u\left(x,0\right)=f(x),\ \text{ on }\ \Omega,\end{aligned}\tag{2.12}$$

where $\Omega\subset\mathbb{R}$, n is the outward normal direction on $\partial\Omega$.

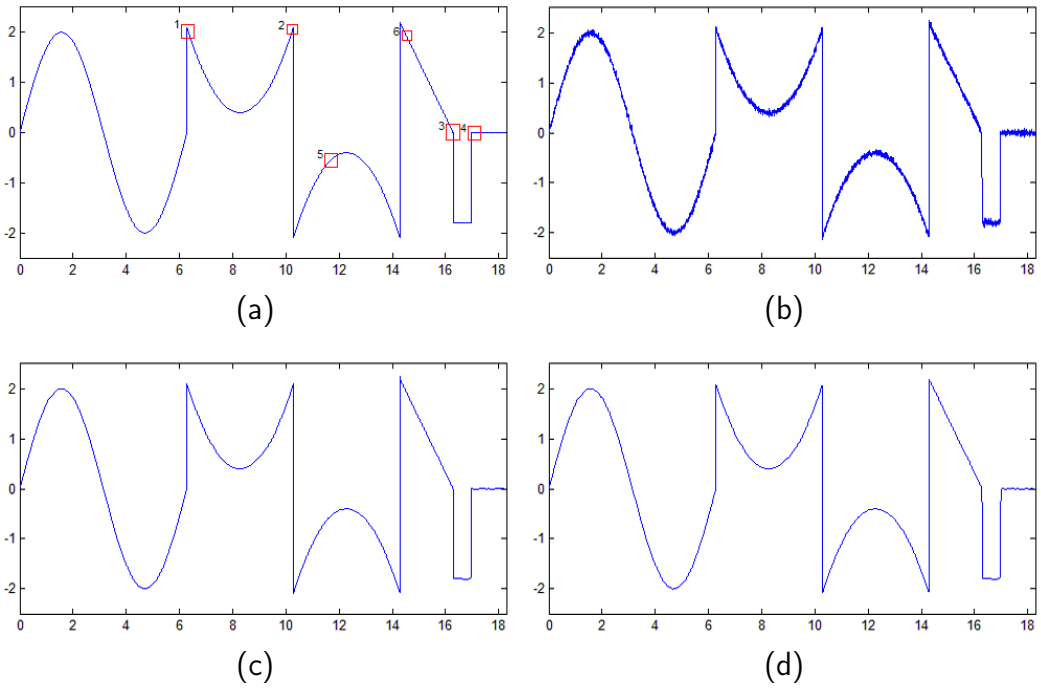

Figure 1: Comparison of adaptive the LLT and the LLT in 1D. (a) Original 1D signal. (b) Noisy 1D signal. (c) Result of the LLT model (2.11). (d) Result of the adaptive LLT model (2.12).

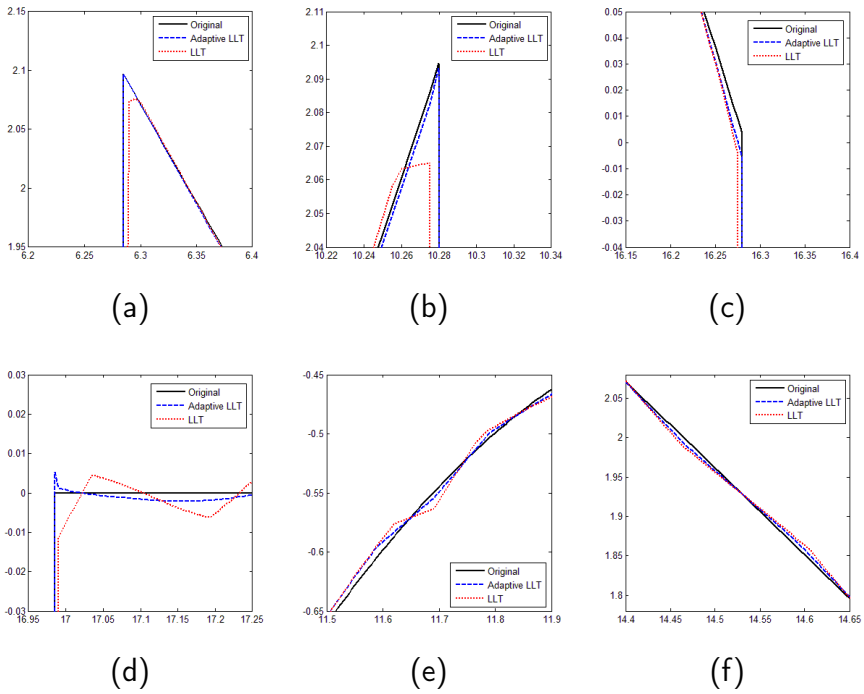

Figure 2: Details of Fig. 1(a). (a) Detail of 1. (b) Detail of 2. (c) Detail of 3. (d) Detail of 4. (e) Detail of 5. (f) Detail of 6.

The numerical implementation of the LLT model (2.11) in this experiment is the finite difference scheme given in paper [2], and we extend it to our adaptive LLT model (2.12). Because it is easy to deduce, we do not give it here.

The test signal is shown in Fig. 1(a), and it includes several jumps and flat lines. The noisy signal is displayed in Fig. 1(b). Denoising results of the LLT model (2.11) and the adaptive LLT model (2.12) are plotted in Fig. 1(c) and Fig. 1(d), respectively. Generally, both models can remove noise. However, from the details of the denoising results (Fig. 2(a) - Fig. 2(d)), all of the jumps are recovered by slop line through the LLT model, and the edges are destroyed. The LLT model can not preserve the edges as the TV model. Not only the LLT model, but also many other fourth-order models can not preserve edges. From theorem 1, the solution of our adaptive LLT model is piecewise planar because of the weighted function $\alpha(f)$. Thus our adaptive LLT model rescues the effect of the LLT model, which can also be demonstrated by Fig. 2(a) - Fig. 2(d).

The equation (2.10) is a fourth-order diffusion equation, which is different from the origianl LLT model because of the feature detection function (2.2). There are many similar improvements [14, 15]. Different from these, we are more concerned about (1.2) rather than (1.3). Because (1.2) has no cross derivative, the model (1.3) is more complicated than model (1.2), and the CFL condition of (1.5) is more stringent than (1.4). Thus, the model (1.3) needs extra computation cost. At the same time, the model (1.4) has strong anisotropic which is necessary for image denoising. Furthermore, the other contribution is the new fourth-order diffusion equation framework based on the new functional.

3 The New Fourth-Order Diffusion Equation

In this section, we present a new fourth-order diffusion framework for image denoising. In this framework, the diffusion coefficient depends on both first-order and second-order derivatives of the image for better preserving edges and other details. Because of making full use of image features, first-order and second-order derivatives, the new model overcomes the weakness that fourth-order PDE models could not preserve structures of the image well.

3.1 The New Fourth-Order Diffusion Equation Framework

Generalizing the fourth-order diffusion equation (2.10), a new fourth-order diffusion equation framework for denoising is given as follows

$$\begin{aligned}
&\frac{\partial u}{\partial t} = -\left(\frac{\Phi(|\nabla u|)}{|u_{xx}|}u_{xx}\right)_{xx} - \left(\frac{\Phi(|\nabla u|)}{|u_{yy}|}u_{yy}\right)_{yy}, \ \text{in } \Omega\times(0,T)\\
&\frac{\partial u}{\partial \overrightarrow{n}} = 0, \ \text{on } \partial\Omega\times(0,T)\\
&\left(\frac{\Phi(|\nabla u|)}{|u_{xx}|}u_{xx}\right)_{x} n_1 + \left(\frac{\Phi(|\nabla u|)}{|u_{yy}|}u_{yy}\right)_{y} n_2 = 0, \ \text{on } \partial\Omega\times(0,T)\\
&u(x,y,0) = f(x,y), \ \text{on } \Omega
\end{aligned} \tag{3.1}$$

where $\frac{\Phi(|\nabla u|)}{|u_{xx}|}, \frac{\Phi(|\nabla u|)}{|u_{yy}|}$ are diffusion coefficients, and $\Phi(|\nabla u|)$ is an adaptive coefficient. The boundary condition is similar to (2.9), and initial value is selected as the noisy

image. In our new fourth-order diffusion equation framework (3.1), the diffusion behavior dominated by the diffusion coefficients. For better smoothing noise and preserving structure, the image feature information should be fully utilized to design the diffusion coefficients, which is the reason why diffusion coefficients, $\frac{\Phi(|\nabla u|)}{|u_{xx}|}$ and $\frac{\Phi(|\nabla u|)}{|u_{yy}|}$, depend on both first-order and second-order derivatives. On the one hand, because of the second-order dirivative part, our framework could overcome staircase effect. On the other hand, the first-order derivative can better detect edges even with strong noise, so $\Phi(|\nabla u|)$ plays a role in preserving structures in our framework.

About the adaptive coefficient $\Phi(|\nabla u|)$, it is $\alpha(f)$ in equation (2.10). In the framework (3.1), we replace $\alpha(f)$ with more general edge detection function. $\alpha(f)$ only depends on the Gaussian convolution of noisy image. But, with the evolution of the fourth-order equation, the result u at time t is more precious than $G * f$. Therefore, we consider a function of u rather than f to measure the degree of jumps. More efficient edge detection functions are exploied in the next section.

We choose the noisy image f as the initial value. In addition, there is no source term $-\lambda(u-f)$ in our new fourth-order diffusion equation framework (3.1), which is equivalent to no fidelity term $\lambda \int_\Omega (u-f)^2 \, \mathrm{d}x\mathrm{d}y$ in the corresponding functional. As indicated in [28] and [29], the parameter λ mostly decide the influence of source term which controls the fidelity of the solution to the input image, however, the parameter λ is susceptible to the noise. Thus, how to choose the parameter λ for an excepted result in the experiment is not easy. Moreover, in the theory of nonlinear diffusion equation, adding a fidelity term results in the resultant image close to the original image which may cause the noise not sufficiently removed. Therefore, we take the source term $-\lambda(u-f)$ away to make the model easy to analyse and reduce the number of parameters.

3.2 The Proposed Model

We recommed two adaptive coefficients:

$$\Phi_a = \frac{1}{\sqrt{1 + \left(|\nabla G_\sigma * u| / K\right)^2}}.$$

and

$$\Phi_b = \frac{1}{1 + \left(|\nabla G_\sigma * u| / K\right)^2},$$

where K is a parameter. As shown in Fig. 3(a), with same K, mapping functions of Φ_a and Φ_b are different, and Φ_b is more sensitive to edges compared with Φ_a, which leads to subtle differences in the denoising results. However, these options have a common feature in general. Φ_a and Φ_b are all bounded between 0 and 1. In the points at the edge, adaptive coefficient Φ attends to 0 for reducing the smoothing and protecting jump, and Φ attend to 1 for smoothing noise in the flat area. That is the same as the original intention introduced in subsection 2.1 and we generalize it using evolving result u instead of noise image f. From Fig. 3(b), the sensitive interval of Φ_a to $|\nabla G * u|$ is different with different K. Specifically, the reaction interval of Φ_a is becoming bigger, as the increase of K. Φ_b also has the same property about K. Note that all of the coefficients in the above options are related to the first-order derivative, and the reason for this is detailed in the next subsection.

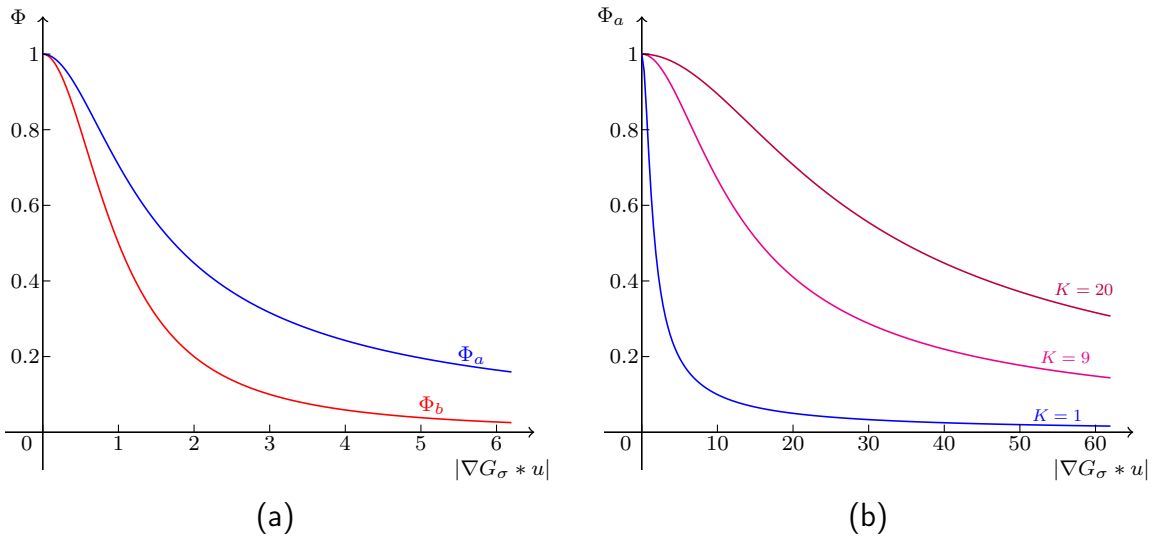

Figure 3: Analysis of Φ_b and Φ_a. (a) Mapping of Φ_b and Φ_a, $K = 1$. (b) Mapping of Φ_a with different K.

Both Φ_a and Φ_b detect the edge of the image by $|\nabla G_\sigma * u|$. The reason for convoluting image u with Gaussian function is inhibiting the influence of noise and obtaining a more precious edge. Because the adaptive coefficient would self-correct during the evolution process, utilizing more accurate coefficient makes the diffusion process less 'detours'. Thus, the evolution rate would be accelerated with accurate edge detection results, $|\nabla G_\sigma * u|$, compared with $|\nabla u|$.

In conclusion, the proposed models are as follows,

$$\begin{aligned}
&\frac{\partial u}{\partial t} = -\left(\Phi_a \frac{u_{xx}}{|u_{xx}|}\right)_{xx} - \left(\Phi_a \frac{u_{yy}}{|u_{yy}|}\right)_{yy}, \ \text{in } \Omega \times (0,T) \\
&\frac{\partial u}{\partial \overrightarrow{n}} = 0, \ \text{on } \partial\Omega \times (0,T) \\
&\left(\Phi_a \frac{u_{xx}}{|u_{xx}|}\right)_x n_1 + \left(\Phi_a \frac{u_{yy}}{|u_{yy}|}\right)_y n_2 = 0, \ \text{on } \partial\Omega \times (0,T) \\
&u(x,y,0) = f(x,y), \ \text{on } \Omega.
\end{aligned} \tag{3.2}$$

and

$$\begin{aligned}
&\frac{\partial u}{\partial t} = -\left(\Phi_b \frac{u_{xx}}{|u_{xx}|}\right)_{xx} - \left(\Phi_b \frac{u_{yy}}{|u_{yy}|}\right)_{yy}, \ \text{in } \Omega \times (0,T) \\
&\frac{\partial u}{\partial \overrightarrow{n}} = 0, \ \text{on } \partial\Omega \times (0,T) \\
&\left(\Phi_b \frac{u_{xx}}{|u_{xx}|}\right)_x n_1 + \left(\Phi_b \frac{u_{yy}}{|u_{yy}|}\right)_y n_2 = 0, \ \text{on } \partial\Omega \times (0,T) \\
&u(x,y,0) = f(x,y), \ \text{on } \Omega.
\end{aligned} \tag{3.3}$$

3.3 Edge Preservation

The choice of diffusion equation in the fourth-order PDE-based denoising method has been discussed. The opinion of [18] proposed by Hajiaboli is that using a function

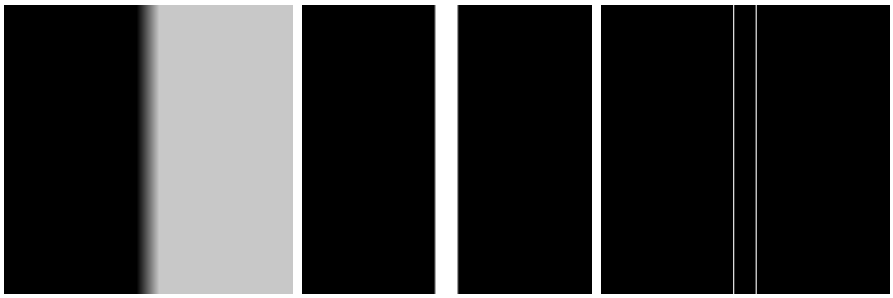

Figure 4: Comparison of edge detection operators in synthetic image. (a) Synthetic image. (b) Gradient of synthetic image. (c) Laplace of synthetic image.

of the gradient modulus is more proper than Laplace map of the image. In this subsection, we detailedly analyse the two categories of edge detection operators, gradient and Laplace. Because we apply the edge detection to the diffusion coefficient, directions of the edge are not discussed here. The following are two specific operator expressions. The modulus of the gradient is

$$|\nabla f(x,y)| = \sqrt{\left(\frac{\partial f(x,y)}{\partial x}\right)^2 + \left(\frac{\partial f(x,y)}{\partial y}\right)^2},$$

and the absolute value of the Laplacian is

$$|\Delta f(x,y)| = \left|\frac{\partial^2 f(x,y)}{\partial x^2} + \frac{\partial^2 f(x,y)}{\partial y^2}\right|.$$

Let's compare the edge detection capabilities of the gradient operator and the Laplace operator (Fig. 4 and Fig. 5). It should be noted that the gradients and Laplaces in Fig. 4 and Fig. 5 are all normalized to $[0, 1]$. Fig. 4(a) is a synthetic image with a weak edge. We can easily see that the gradient operator could successfully detect the weak edges. However, the result of Laplace is two margin of weak edge. The Laplace operator has a bilateral effect and can't do anything about weak edges. Secondly, a natural image is taken for testing in Fig. 5, and the edges detected by the gradient operator have a stronger contrast than Laplace's. The second row in Fig. 5 is a noisy situation. When the standard deviation of noise is 20, the Laplace operator will lose the ability to detect the edge, yet we could find edges easily in Fig. 5(e). For the Laplace operator, the response to isolated pixels is stronger than the edge or line. The above confirms the view that the Laplace operator is sensitive to the noise and only applicable to noise-free images.

Form the above analysis, we can know that the gradient function is better than the Laplace operator in edge detection. The advantages include three aspects: (a) the ability of weak edge detection, (b) the high intensity of detected edge, and (c) the anti-noise performance. In conclusion, gradient function is preferable and suitable to detect edges. That is the reason why we choose functions with gradient in section 3.2.

Images shown in Fig. 6 are coefficients of the LLT model and our proposed model. The images of the first row are to original pepper image Fig. 5(a), and the

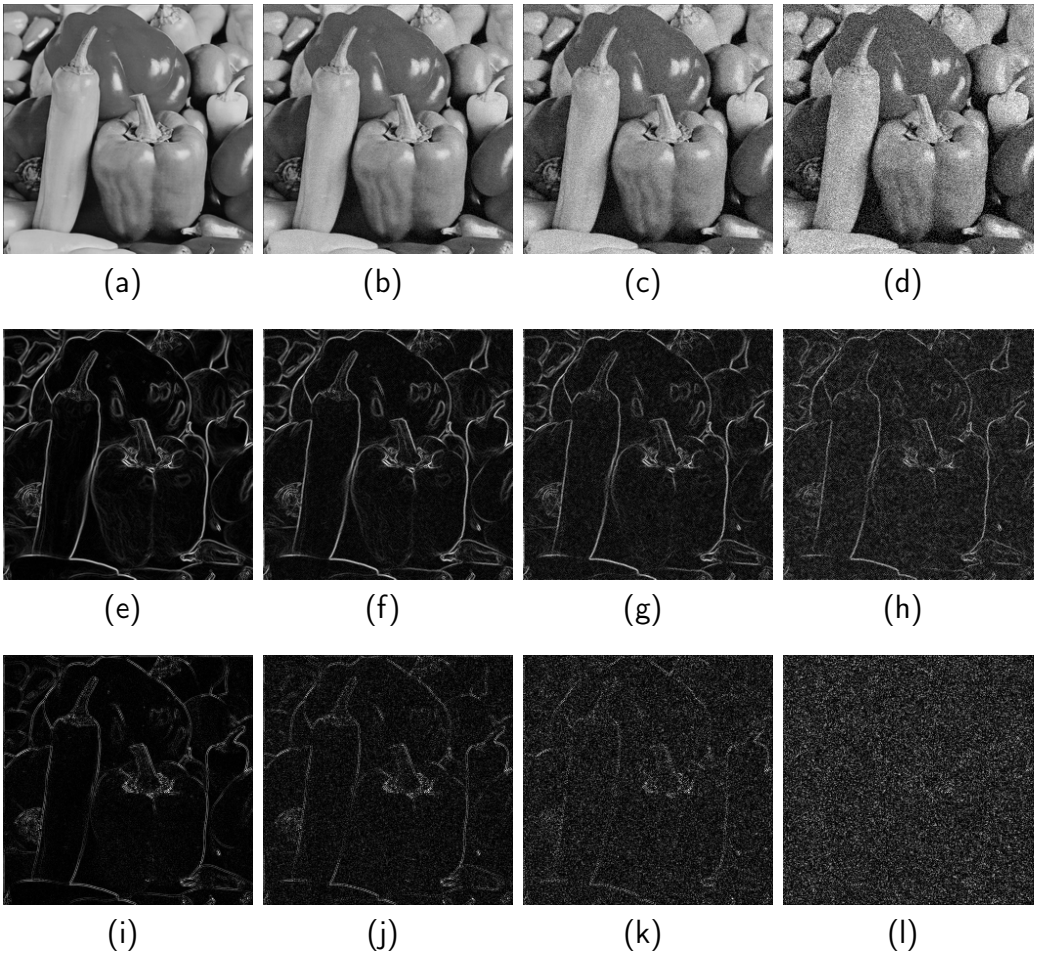

Figure 5: Comparison of edge detection operators in natural image. (a) Original image. (b) Noisy, $\sigma_{\mathrm{n}} = 5$. (c) Noisy, $\sigma_{\mathrm{n}} = 10$. (d) Noisy, $\sigma_{\mathrm{n}} = 20$. (e) Gradient of original image. (f) Gradient of noisy image, $\sigma_{\mathrm{n}} = 5$. (g) Gradient of noisy image, $\sigma_{\mathrm{n}} = 10$. (h) Gradient of noisy image, $\sigma_{\mathrm{n}} = 20$. (i) Laplace of noisy image. (j) Laplace of noisy image, $\sigma_{\mathrm{n}} = 5$. (k) Laplace of noisy image, $\sigma_{\mathrm{n}} = 10$. (l) Laplace of noisy image, $\sigma_{\mathrm{n}} = 20$.

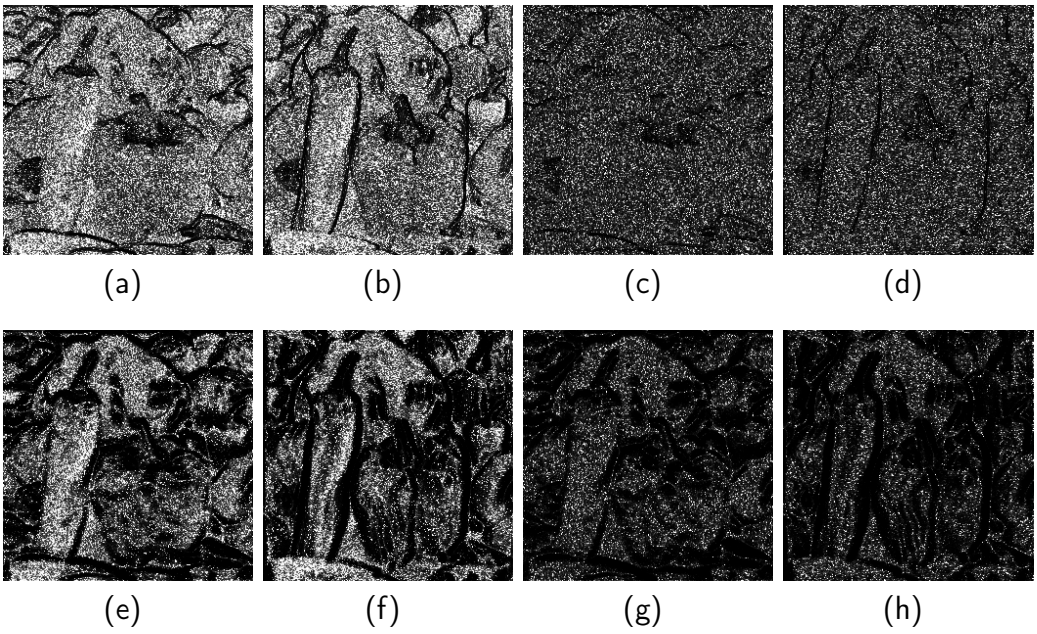

Figure 6: Comparison of diffusion coefficients of proposed model and the LLT model. $\Phi = \Phi_b$, $K = 1$. (a) $\frac{1}{|u_{xx}|}$. (b) $\frac{1}{|u_{yy}|}$. (c) $\frac{1}{|u_{xx}|}$, $\sigma_{\mathrm{n}} = 5$. (d) $\frac{1}{|u_{yy}|}$, $\sigma_{\mathrm{n}} = 5$. (e) $\frac{\Phi_b}{|u_{xx}|}$. (f) $\frac{\Phi_b}{|u_{yy}|}$. (g) $\frac{\Phi_b}{|u_{xx}|}$, $\sigma_{\mathrm{n}} = 5$. (h) $\frac{\Phi_b}{|u_{yy}|}$, $\sigma_{\mathrm{n}} = 5$.

second row are to noisy image whose noise level is 5. In these images, the higher the grey value, the larger the coefficient. On the one hand, these images indicate that the coefficients of the LLT model are generally larger than the coefficients of our model. In other words, Fig. 6(a) - Fig. 6(d) seem more brightness than Fig. 6(e) - Fig.6(h). On the other hand, the edges in the coefficient images of the LLT model are thinner. It illustrates that our model processes edges and other areas more differently. The noisy case has the same phenomenon. Therefore, putting the first-order derivative to the original second-order diffusion coefficients is helpful to restore edges and structures.

4 Numerical Implementation

For numerical solving the proposed diffusion equation (3.1), we design two different finite difference schemes, finite difference explicit scheme and semi-implicit scheme, and develop fast explicit diffusion algorithm (FED) and additional operator splitting algorithm (AOS) to reduce computational efficiency, respectively.

4.1 The Finite Difference Explicit Scheme

Assuming space grid size of h, $h = 1$ for digital image, and the space discretization mesh is

$$x = ih, \quad i = 0, 1, ...I$$
$$y = jh, \quad j = 0, 1, ...J.$$

where $I \times J$ is the size of image support. The time step by is τ, $t = n\tau$, $n = 0, 1, ...$ Thus $u_{i,j}^n = u(ih, jh, n\tau)$. The finite diffusion explicit scheme is given as follows,

$$u_{i,j}^{n+1} = u_{i,j}^n - \tau \left[D_{xx} \left(\Phi_x \left((u_\sigma)_{i,j}^n \right) \alpha_\epsilon \left(u_{i,j}^n \right) D_{xx} u_{i,j}^n \right) + D_{yy} \left(\Phi_y \left((u_\sigma)_{i,j}^n \right) \beta_\epsilon \left(u_{i,j}^n \right) D_{yy} u_{i,j}^n \right) \right], \quad (4.1)$$

where

$$\begin{aligned} D_{xx}\left(u_{i,j}\right) &= \left(u_{i-1,j} + u_{i+1,j} - 2u_{i,j}\right)/h^2, \\ D_{yy}\left(u_{i,j}\right) &= \left(u_{i,j-1} + u_{i,j+1} - 2u_{i,j}\right)/h^2, \\ \alpha_\epsilon\left(u_{i,j}\right) &= 1/\left(\left|D_{xx}\left(u_{i,j}\right)\right| + \epsilon\right), \\ \beta_\epsilon\left(u_{i,j}\right) &= 1/\left(\left|D_{yy}\left(u_{i,j}\right)\right| + \epsilon\right), \\ u_\sigma &= G_\sigma * u, \end{aligned}$$

and as in [2], the $\epsilon > 0$ is introduced to avoid zero in the numerical format for the denominator. For the discretization of coefficient Φ, we draw on the discretization scheme of PM equation [7]. When discretize x direction, data in the y direction is not considered. It is same as the discretization of y direction. Thus the scheme is more anisotropic. When the efficient is taken as Φ_a, the discretizations are

$$\begin{aligned} \Phi_x\left((u_\sigma)_{i,j}\right) &= 1 \Bigg/ \sqrt{1 + \left(\frac{(u_\sigma)_{i+1,j} - (u_\sigma)_{i-1,j}}{2h} \right)^2}, \\ \Phi_y\left((u_\sigma)_{i,j}\right) &= 1 \Bigg/ \sqrt{1 + \left(\frac{(u_\sigma)_{i,j+1} - (u_\sigma)_{i,j-1}}{2h} \right)^2}. \end{aligned}$$

and when it is Φ_b,

$$\begin{aligned} \Phi_x\left((u_\sigma)_{i,j}\right) &= 1 \Bigg/ \left(1 + \left(\frac{(u_\sigma)_{i+1,j} - (u_\sigma)_{i-1,j}}{2h} \right)^2 \right), \\ \Phi_y\left((u_\sigma)_{i,j}\right) &= 1 \Bigg/ \left(1 + \left(\frac{(u_\sigma)_{i,j+1} - (u_\sigma)_{i,j-1}}{2h} \right)^2 \right), \end{aligned}$$

And the discretization of initial condition is $u_{i,j}^0 = f_{i,j}$. To incorporate the boundary conditions, the approximations at the boundaries need to be treated properly. We summarized as follows,

$$\begin{aligned} u_{i-1,j} &= u_{0,j} \quad \text{if } i = 0, \\ u_{i+1,j} &= u_{I,j} \quad \text{if } i = I, \\ u_{i,j-1} &= u_{i,0} \quad \text{if } j = 0, \\ u_{i,j+1} &= u_{i,J} \quad \text{if } j = J, \end{aligned}$$

and

$$\begin{aligned} \Phi_x(u_{i-1,j}^n)\alpha_\epsilon(u_{i-1,j}^n)D_{xx}u_{i-1,j}^n &= \Phi_x(u_{0,j}^n)\alpha_\epsilon(u_{0,j}^n)D_{xx}u_{0,j}^n \quad \text{if } i = 0, \\ \Phi_x(u_{i+1,j}^n)\alpha_\epsilon(u_{i+1,j}^n)D_{xx}u_{i+1,j}^n &= \Phi_x(u_{I,j}^n)\alpha_\epsilon(u_{I,j}^n)D_{xx}u_{I,j}^n \quad \text{if } i = I, \\ \Phi_y(u_{i,j-1}^n)\beta_\epsilon(u_{i,j-1}^n)D_{yy}u_{i,j-1}^n &= \Phi_y(u_{i,0}^n)\beta_\epsilon(u_{i,0}^n)D_{yy}u_{i,0}^n \quad \text{if } j = 0, \\ \Phi_y(u_{i,j+1}^n)\beta_\epsilon(u_{i,j+1}^n)D_{yy}u_{i,j+1}^n &= \Phi_y(u_{i,J}^n)\beta_\epsilon(u_{i,J}^n)D_{yy}u_{i,J}^n \quad \text{if } j = J. \end{aligned}$$

Furthermore, we introduce a notation $U^n \in \mathbb{R}^{IJ}$ which represents a vector containing the values at each pixel when the time level is n and $E \in \mathbb{R}^{IJ \times IJ}$ is the unit matrix. The explicit shceme of matrix form can be written as:

$$U^{n+1} = \left(E + \tau A\left(U^n\right)\right) U^n. \tag{4.2}$$

There is no calculation of matix inversion, so that it's easy in each iteration.

Unfortunately, this explicit scheme (4.1) has a serious drawback that the time step permitted has a strict limit for stability reasons due to the high nonlinearity and stiffness. So the number of iterations evolving to the target image is too large, which renders the numerical scheme inefficient. Therefore, it is desirable to find more efficient tools to accelerate the implementation of our model. One possibility is the Fast Explicit Diffusion (FED) scheme.

4.2 FED Scheme

In this section, in order to solve the problem of low efficiency of explicit format (4.1), FED algorithm that could significantly improve computing speed is applied to accelerate calculation.

FED has been presented and investigated by Grewenig et al. [30] in 2010, was further elaborated in 2013 [31] and is a widely applicable algorithm. FED scheme is similar to the method under the name Super Time Stepping (STS) by Gentzsch et al.[32][33]. FED is a slight modification of the explicit finite difference discretisation, which can boost the efficiency of the explicit scheme by several orders of magnitude. The way that FED promotes the calculation speed is that they perform cycles of explicit diffusion schemes with varying time step sizes. Since within each cycle up to 50 percent of all steps may violate the stability condition, one can achieve very large diffusion times.

In the following, the algorithm of FED schemes was generalised in a straightforward way to our explicit scheme. After we obtain the equations (4.2) representing the explicit scheme from the above deduction, a cycle of varying time step size τ_i needs to be calculated,

$$\tau_i = \frac{2}{\mu_{\max}\left(A\right)} \cdot \frac{1}{2\cos^2\left(\pi \cdot \frac{2i+1}{4n+2}\right)}, \quad i = 0, 1, 2, ..., n_c - 1, \tag{4.3}$$

where $\mu_{\max}$ represents the largest modulus of the eigenvalues of A, n_c is the number of iteration in each cycle and A corresponds to the $A\left(U^n\right)$ in (4.2). The corresponding total time of this cycle is

$$t_{n_c} = \sum_{i=0}^{n_c-1} = \frac{h^2}{2} \begin{pmatrix} n_c + 1 \\ 2 \end{pmatrix}. \tag{4.4}$$

Because the proposed models are nonlinear equations, the matrix A changes depending on U^n. In order to apply cyclic schemes, in the same way as [28], we use a priori estimate for the values $\mu_{\max}$ for the numerical stability. Using the Gershgorin's theorem in [34], one can easily obtain that $\mu_{\max}\left(A\right) \leqslant \frac{32}{\epsilon}$.

It can be proved that the scheme is stable in each complete cycle. From (4.3), it is obvious that near half of the time step size τ_i in (4.3) far exceeds the stability condition for the explicit scheme with a constant time step size [31], and that is the reason of FED could significantly improve computational efficiency.

The FED algorithm of our model can be expressed as Algorithm 1.

Algorithm 1 FED Algorithm of Proposed Models

Require: Original image f, stopping time T, number M of outer FED cycles, new fourth order model's parameters.

1: Compute the samllest n_c how to calculate n_c so that the stopping time t_{n_c} of one FED cycle fulfils $t_{n_c} \geq T/M$, and define $q = T/\left(M \cdot t_{n_c}\right)$.
2: Compute the time step sizes $\widetilde{\tau}_i = q \cdot \tau_i$ according to (4.3).
3: Choose a suitable ordering for the step size $\widetilde{\tau}_i$ according to [35] for numerical stability reasons.
4: Initialize $U^{0,0} = f$.
5: **while** $k = 0, 1, ..., M-1$ **do**
6: **for** $i = 0, 1, ..., n_c - 1$ **do**
7: Calculate the corresponding matrix $A\left(U^{k,i}\right)$.
8: Compute $U^{k,i+1} = \left(I + \widetilde{\tau}_i A\left(U^{k,i}\right)\right) U^{k,i}$.
9: **end for**
10: $U^{k+1,0} = U^{k,n_c}$.
11: **end while**

4.3 The Semi-Implicit Scheme

In this subsection, we give a semi-implicit scheme,

$$u_{i,j}^{n+1} = u_{i,j}^{n} - \tau\left[D_{xx}\left(\Phi_x\left((u_\sigma)_{i,j}^{n}\right)\alpha_\epsilon\left(u_{i,j}^{n}\right)D_{xx}u_{i,j}^{n+1}\right) + D_{yy}\left(\Phi_y\left((u_\sigma)_{i,j}^{n}\right)\beta_\epsilon\left(u_{i,j}^{n}\right)D_{yy}u_{i,j}^{n+1}\right)\right]. \tag{4.5}$$

Boundary condition is as same as the one of explicit scheme (2.9).

Next, we give a theorem to prove the unconditional stability of the semi-implicit scheme (4.5) using Fourier method.

Theorem 2. *The semi-emplicit scheme (4.5) is unconditionally stable.*

Proof. Setting $(F_x)_{i,j}^{n} = \Phi_x\left((u_\sigma)_{i,j}^{n}\right)\alpha_\epsilon\left(u_{i,j}^{n}\right)$, $(F_y)_{i,j}^{n} = \Phi_y\left((u_\sigma)_{i,j}^{n}\right)\beta_\epsilon\left(u_{i,j}^{n}\right)$, the scheme (4.5) is rewritten as

$$u_{i,j}^{n+1} = u_{i,j}^{n} - \tau\left[D_{xx}\left((F_x)_{i,j}^{n}D_{xx}u_{i,j}^{n+1}\right) + D_{yy}\left((F_y)_{i,j}^{n}D_{yy}u_{i,j}^{n+1}\right)\right].$$

The semi-implicit scheme can be expanded as

$$\begin{aligned}
u_{i,j}^{n+1} + \frac{\tau}{h^4}\Big((F_x)_{i-1,j}^{n}u_{i-2,j}^{n+1} + \left(-2(F_x)_{i,j}^{n} - 2(F_x)_{i-1,j}^{n}\right)u_{i-1,j}^{n+1} \\
+ \left((F_x)_{i-1,j}^{n} + (F_x)_{i+1,j}^{n} + 4(F_x)_{i,j}^{n}\right)u_{i,j}^{n} \\
+ \left(-2(F_x)_{i,j}^{n} - 2(F_x)_{i+1,j}^{n}\right)u_{i+1,j}^{n+1} + (F_x)_{i+1,j}^{n}u_{i+2,j}^{n+1}\Big) \\
+ \frac{\tau}{h^4}\Big((F_y)_{i,j-1}^{n}u_{i,j-2}^{n+1} + \left(-2(F_y)_{i,j}^{n} - 2(F_y)_{i,j-1}^{n}\right)u_{i,j-1}^{n+1} \\
+ \left((F_y)_{i,j-1}^{n} + (F_y)_{i,j+1}^{n} + 4(F_y)_{i,j}^{n}\right)u_{i,j}^{n} \\
+ \left(-2(F_y)_{i,j}^{n} - 2(F_y)_{i,j+1}^{n}\right)u_{i,j+1}^{n+1} + (F_y)_{i,j+1}^{n}u_{i,j+2}^{n+1}\Big) = u_{i,j}^{n}
\end{aligned} \tag{4.6}$$

Next, we try to verify that the semi-implicit scheme (4.5) is unconditionally stable using Fourier method (introduced in many numerical analysis books). Set

$(F_x)_{i,j}^n = \bar{F}_x$ and $(F_y)_{i,j}^n = \bar{F}_y$, then (4.6) becames

$$u_{i,j}^{n+1} + \frac{\tau}{h^4}\left(\bar{F}_x u_{i-2,j}^{n+1} - 4\bar{F}_x u_{i-1,j}^{n+1} + 6\bar{F}_x u_{i,j}^{n} - 4\bar{F}_x u_{i+1,j}^{n+1} + \bar{F}_x u_{i+2,j}^{n+1}\right) + \frac{\tau}{h^4}\left(\bar{F}_y u_{i,j-2}^{n+1} - 4\bar{F}_y u_{i,j-1}^{n+1} + 6\bar{F}_y u_{i,j}^{n} - 4\bar{F}_y u_{i,j+1}^{n+1} + \bar{F}_y u_{i,j+2}^{n+1}\right) = u_{i,j}^{n}.$$

Perform Fourier analysis on both ends of the above equation,

$$\hat{U}^{n+1}(k)\left(1 + \frac{\tau}{h^4}\bar{F}_x\left(e^{-i2kh} - 4e^{-ikh} + 6 - 4e^{ikh} + e^{i2kh}\right) + \frac{\tau}{h^4}\bar{F}_y\left(e^{-iI2kh} - 4e^{-iIkh} + 6 - 4e^{iIkh} + e^{i2Ikh}\right)\right) = \hat{U}^{n}(k),$$

where the i in above equation is an imaginary unit. And the growth factor is

$$\begin{aligned} G(\tau,k) &= 1 \Big/ \left(1 + \frac{\tau}{h^4}\bar{F}_x C_x + \frac{\tau}{h^4}\bar{F}_y C_y\right), \\ C_x &= \left(e^{-i2kh} - 4e^{-ikh} + 6 - 4e^{ikh} + e^{i2kh}\right), \\ C_y &= \left(e^{-iI2kh} - 4e^{-iIkh} + 6 - 4e^{iIkh} + e^{i2Ikh}\right). \end{aligned}$$

Therefore,

$$\begin{aligned} C_x &= 6 + 2\cos(2kh) - 8\cos(kh) \\ &= 4\left(\cos(kh) - 1\right)^2 \geq 0. \end{aligned}$$

With the same reason we have

$$C_y = 4\left(\cos(Ikh) - 1\right)^2 \geq 0.$$

With $\bar{F}_x > 0$ and $\bar{F}_y > 0$, we get $G(\tau,k) \leq 1$. This semi-implicit scheme is unconditional stable that (4.5) satisfieds all discrete scale-space requirements for all time step size $\tau > 0$. □

Theorem 2 shows that the stability of our numerical scheme (4.5) is not subjected to the step size τ. Thus, using this semi-implicit scheme, we could increase the computational efficiency by increasing the step size.

Different from (4.1), numerical discretization of diffusion main items at semi-implicit (4.5) employs the $(n+1)^{th}$ time level values. This scheme dosen't give the solution $u_{i,j}^{n+1}$ directly, hence, we have to solve a linear system first.

The semi-implicit scheme results in a linear simultaneous equations that could be expressed in matrix vector form can be rewritten as

$$U^{n+1} = U^n + \tau A^n U^{n+1}, \tag{4.7}$$

where A^n is sparse, and the solution of (4.7) is formally given by

$$U^{n+1} = \left(E - \tau A^n\right)^{-1} U^n. \tag{4.8}$$

Compute the linear system (4.8) is expensive [36, 37]. In the 2-D case, the matrix A^n reveal a larger banwidth, and from (4.6) the matrix is a nine diagonal one. By observing the formula (4.8), we find that we have to solve a large matrix whose dimension is $IJ \times IJ$. For higher dimensions, the computational effort could be more massive. Using direct algorithms such as Gaussian elimination or Jacobian

iteration would lead to large storage and computation effort. Moreover, when τ is larger, the condition number of the system matrix is larger, so, classical iterative algorithms will be very slow. Therefore, it is desirable to find more efficient tools to accelerate the implementation of our scheme.

In the next section, we focus on a splitting-based alternative. Additional operator splitting (AOS) algorithm is a possible method. It is simple and doesn't require to specify any additional parameters.

4.4 AOS Scheme

In this section, we apply a convenient and efficient algorithm, additional operator splitting (AOS)[36, 37, 38], to solve the semi-implicit scheme (4.5). The reason for improving the computing efficiency is that transform a high-dimensional problem into several one-dimensional problems. That could directly avoid calculation of complicated matrix. Different from traditional multiplicative splittings such as ADI [39, 40, 41, 42], LOD [43, 44, 45, 46] or D'yakonov splitting, all axes are treated in the same manner, and additional possibilities for efficient realizations on parallel and distributed architectures appear. Under typical accuracy requirements, AOS schemes are ten times more efficient than the semi-implicit schemes at least.

The following we directly apply AOS method to the semi-implicit (4.7). That could be divided into two sub-steps. First, we do one-dimensional diffusion of rows and columns, separately, and get two intermediate results U_1^{n+1} and U_2^{n+1}. Set A_x is rows processing coefficient matrix, and A_y is columns matrix. What needs to be calculated is that:

$$\left(E - 2\tau A_x^n\right) U_1^{n+1} = U^n, \tag{4.9}$$

$$\left(E - 2\tau A_y^n\right) U_2^{n+1} = U^n. \tag{4.10}$$

At the last, calculate the average of the two as a result of a complete iteration,

$$U^{n+1} = \frac{U_1^{n+1} + U_2^{n+1}}{2}.$$

The AOS scheme of (4.8) is

$$U^{n+1} = \frac{\left[\left(E - 2\tau A_x^n\right)^{-1} + \left(E - 2\tau A_y^n\right)^{-1}\right] U^n}{2}.$$

Since the final result U^{n+1} is independent of the calculation order of U_1^{n+1} and U_2^{n+1}, this algorithm satisfies rotation invariance.The dimension solving (4.9) and (4.10) is $IJ \times IJ$,and the matrix calculated is five diagonal. Even if that, in the pratical solving process, we comput by row (column). For (4.9), it is composed of I independent sub-eqations, and each of the sub-equations is a linear simultaneous equations whose elements are columns of U_1^{n+1}. Hence, just need to solve the equations whose dimension is I and J. For example, the sub-equations of i-th row in (4.9) could be written as

$$\left(E - 2\tau A_{x,i}^n\right) U_{1,i}^{n+1} = \left(U_x^n\right)_i,$$

where $A_{x,i}^n$ represents the i-th row of A_x^n, $U_{1,i}^{n+1}$ represents the i-th row of U_1^{n+1}, and $\left(U_x^n\right)_i$ is the i-th row of U^n. This could also reduce memory consumption in the

computer numerical operation. The calculation of (4.10) is same as (4.9), and we will not repeate here.

In summary, the AOS algorithm could significantly increase computational efficiency while reducing memory costs. The AOS algorithm of the new fourth-order denoising model is summarized as Algorithm 2.

Algorithm 2 AOS Algorithm of Proposed Models

Require: Original image f, number N of iterative, time step size τ, new fourth order model's parameters.

Initialize $u_{i,j}^0 = f_{i,j}$, and arrange $u_{i,j}^0$ into U^0.
for $n = 1, ..., N$ **do**
  Calculate $(F_x)_{i,j}^n$ and $(F_y)_{i,j}^n$, for all $i = 1, 2, ..., I$ and $j = 1, 2, ..., J$.
  for i=1,2,...,I **do**
    Calculate the elements on five diagonals of matrix $\left(E - 2\tau A_{x,i}^n\right)$.
    Solve $U_{1,i}^{n+1}$ from $\left(E - 2\tau A_{x,i}^n\right) U_{1,i}^{n+1} = (U_x^n)_i$.
  end for
  for j=1,2,...,J **do**
    Calculate the elements on five diagonals of matrix $\left(E - 2\tau A_{y,j}^n\right)$.
    Solve $U_{2,j}^{n+1}$ from $\left(E - 2\tau A_{y,j}^n\right) U_{2,j}^{n+1} = \left(U_y^n\right)_j$.
  end for
  Calculate $U^{n+1} = \left(U_1^{n+1} + U_2^{n+1}\right) / 2$.
end for

5 Experiment Results

In this section, we show some experiment results of our algorithms presented in this paper, and compare these models with various classic and recent stat-of-art denoising methods, including TV [1], AFD [8], LLT (1.2) [2], MC [4] and BM3D [47]. Note, in this section, denoising results of the MC model [4] are obtained through multigrid algorithm which is proposed in [5]. Furthermore, we present the results of the explicit scheme (4.1), the FED algorithm (Algorithm 1), and the AOS algorithm (Algorithm 2) with two coefficient options, Φ_a and Φ_b. To simplify the notation, the explicit scheme algorithm using Φ_b is denoted as E-b, and other algorithms are denoted as E-a, FED-b, FED-a, AOS-b, AOS-a. For the evaluations, we have used 8 images shown in Fig. 7, including 2 synthetic images: *geometry* (300×300 pixels, piecewise constant image) and *slope* (128×128 pixels, piecewise planar image), and 6 natural images: *piggy* (341×199 pixels), *butterfly* (256×256 pixels), *castle* (321×481 pixels), *lena* (371×302 pixels), *starfish* (256×256 pixels), and *parrot* (256×256 pixels). Each noisy image is generated by adding gaussian noise to original image according to the additive noise model.

In order to quantify the quality of the denoising results, which is comparing the similarity between the noise-free image and the restored image, we utilize measurements: PSNR (peak signal to noise ratio) and SSIM (structural similarity index). For fairness, all the methods are settled with proper parameters for maximum PSNR [48] and SSIM [49].

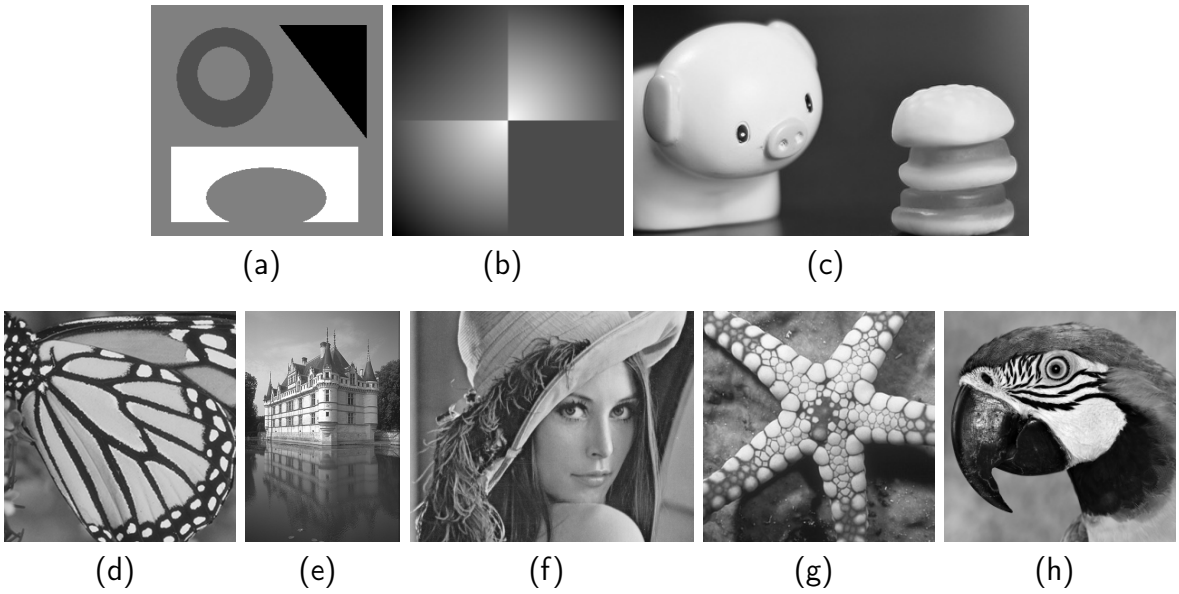

Figure 7: Test images. (a) *geometry*, synthetic image, piecewise constant image. (b) *slope*, synthetic image, piecewise planar image. (c) *piggy*. (d) *butterfly*. (e) *castle*. (f) *lena*. (g) *starfish*. (h) *parrot*.

5.1 Synthetic Images Denoising

In Fig. 8, we first consider the piecewise constant image with five blocks (Fig. 7(a)) which contains different geometric shapes. The original clean image, the noisy image with $\sigma_{\mathrm{n}} = 40$, and the restored images by TV, AFD, LLT, MC, FED-a, and AOS-a are listed. From the results, the noise can be efficiently eliminated by all of the models. In this group of experiments, parameters are $\sigma = 0.5$ and $K = 0.1$ for both FED-a and AOS-a. As expected, the proposed algorithms, FED-a and AOS-a, lead to clearer edges and flatter flat areas. Though there are only terms of x direction and y direction of our proposed discretized scheme, jumps of every direction can be preserved well. In Fig. 9(b), we also show a column of signals of denoised images by LLT, FED-a, AOS-a where the red line is located in Fig. 9(a). Same as the result of the one-dimensional experiment in subsection 2.1, our proposed algorithms can better preserve edges and restore smoother flat areas than the LLT model. In more detail, the restored image of AOS-a is smoother on the basis of preserving edges, and the restored image of FED-a is more sharpen. For TV and AFD edges can be preserved well, but the restored images are not smooth enough in flat areas. And from Fig. 8, the resultant image of AFD are not clean, which contain many speckles around edges. The resultant images of the LLT model and the MC model look patchy in smooth regions and blur in edges. Different from PDE methods, BM3D shows artificial effects, especially around edges. The contrast of the image through BM3D is also changed because the bright irregular shape is darkened. The PSNR and SSIM are shown in Table 1, and the PSNR of FED-a and AOS-a is 1.7dB and 1dB higher than other comparison models at least, respectively. The SSIM of AOS-a is 0.315 higher than other comparison models. For FED-a, the SSIM is 0.005 higher than other models at least. Because there are some speckles in FED-a result, the SSIM is not particularly high as PSNR. These all illustrate the performance of our proposed algorithms.

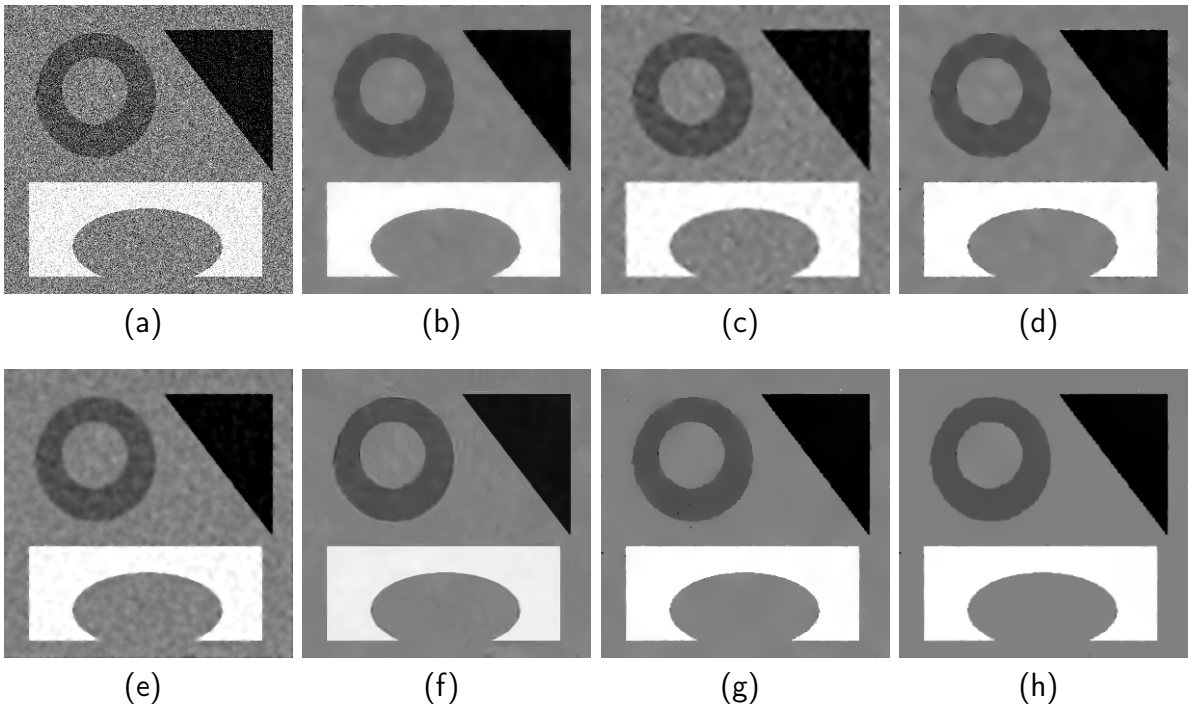

Figure 8: *geometry* (300×300), $\sigma_{\mathrm{n}} = 40$. (a) Noisy, $\sigma_{\mathrm{n}} = 40$. (b) TV. (c) LLT. (d) AFD. (e) MC. (f) BM3D. (g) FED-a. (h) AOS-a.

To further compare the edge preserving ability, another test is performed on the *slope* image, a piecewise planar image with four blocks in Fig. 10. The noise level of the noisy *slope* is 20. The parameters used in FED-a and AOS-a are $\sigma = 0$, $K = 1$. From the results, all of the models can eliminate the noise efficiently. Our algorithms can allow planar, while the TV model yields staircase effects. Note that the results of our algorithm are very smooth in slop areas, which is almost the same as the original image. However, the restored images by TV, AFD, MC, and BM3D are not clean enough and generate edges that do not exist. As discussed before, the LLT model is not good at restoring sharp edges and smooth flat areas. We can also see the difference by the residual images in Fig. 11. The way we get residuals is $f - u + 128$. The residuals from AFD, BM3D and our algorithms are almost uniform which we cannot see the information of the image. However, edges can be vaguely seen from the residuals of TV and MC. Notably, the four edges in the residual from LLT are obvious, which also illustrates the difference between our model and the LLT model.

These two synthetic images experiments (Fig. 8 and Fig. 10) demonstrate that our algorithms can remove noise for smooth results, and also preserve edges well. Our algorithms perform well both in vision and PSNR, SSIM indicators. In fact, from Theorem 1, the result of the adaptive LLT model without fidelity term is piecewise planar, and our fourth-order diffusion equation is a generalization of the adaptive LLT model (2.1). Thus, out algorithms perform well both in flat area smoothing and edge preserving.

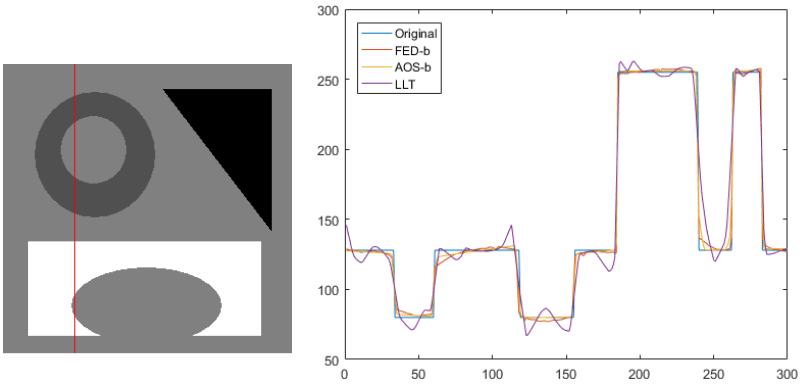

Figure 9: Plot of a column signal. (a) Location of the signal. (b) Plots of signals of original image and denoised images by LLT, FED-a, AOS-a.

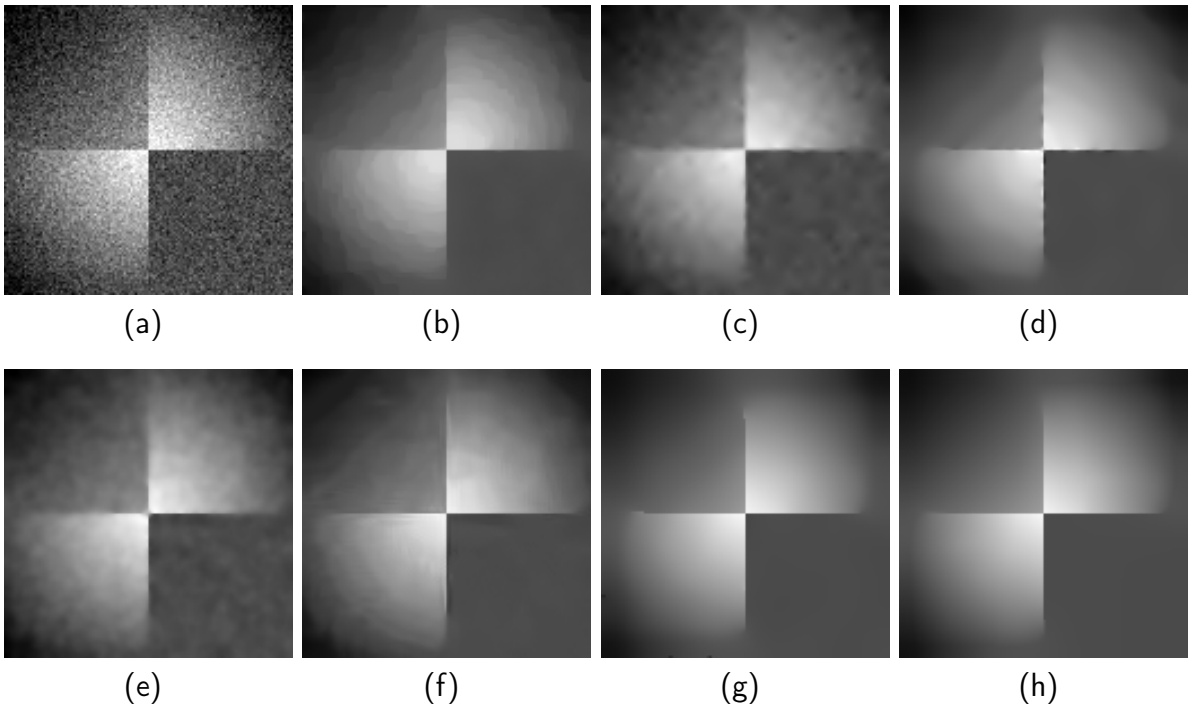

Figure 10: *slope* (128×128), $\sigma_{\mathrm{n}} = 20$. (a) Noisy, $\sigma_{\mathrm{n}} = 20$. (b) TV. (c) LLT. (d) AFD. (e) MC. (f) BM3D. (g) FED-a (h) AOS-a

Table 1: PSNR (dB) and SSIM results of denoising results of synthetic images by different methods.

$\sigma_{\mathbf{n}}$	**Images**	**TV**	**LLT**	**AFD**	**MC**	**BM3D**	**FED-a**	**AOS-a**
40	*geometry*	32.873	27.400	32.343	29.691	28.691	34.831	34.147
		0.188	0.145	0.190	0.168	0.186	0.202	0.512
20	*slope*	37.158	33.023	38.900	34.513	38.918	42.080	43.221
		0.714	0.640	0.764	0.631	0.712	0.802	0.863

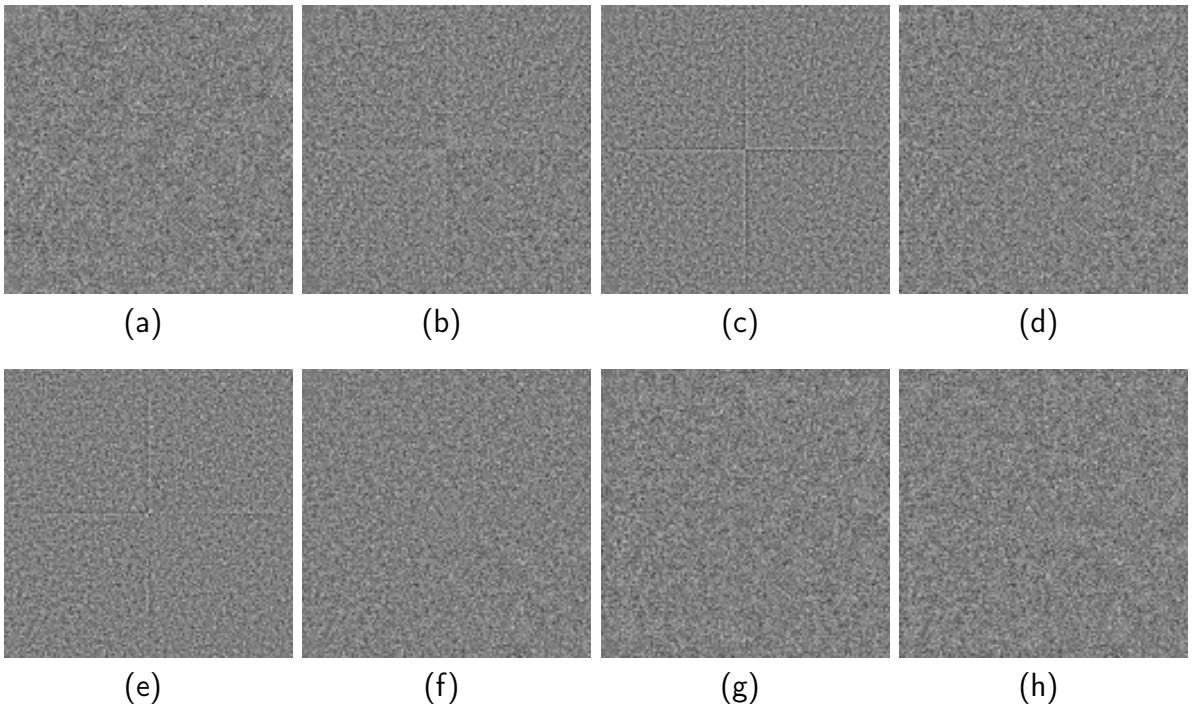

Figure 11: Difference of denoised *slope* (128×128), $\sigma_{\mathrm{n}} = 20$. (a) Noise added to *slope*, $\sigma_{\mathrm{n}} = 20$. (b) TV. (c) LLT. (d) AFD. (e) MC. (f) BM3D. (g) FED-a (h) AOS-a

5.2 Parameter Setting and Analysis

In our algorithms, there are several parameters, mainly step size τ, σ, K, number of iteration N, and ϵ for avoiding zero in the denominator. For explicit scheme, the step size τ needs to be set to a small value in order to guarantee the stability. To be on the safe side, we set $\tau = 0.01$ in our explicit scheme with Φ_a and Φ_b, and experiments illustrate the stability of the explicit scheme. For FED algorithm (Algorithm 1), we fix $T = 2M$ for stability and convenient. Also, experiments show that FED can still accelerate the algorithm in this case from Table 4. More relaxed parameters τ, T and M can also guarantee the stability. However, because of the unconditional stability from Theorem 2, the step size τ in the semi-implicit scheme can be taken arbitrarily. In our experiments, we set $\tau = 2$ for Algorithm 2.

Now, let us explore the influence of parameters σ and K to the models. Firstly, we want to illustrate that out models can yield good results with a great range of σ and K. For this purpose, we conduct parametric experiments for E-a where parameter σ belongs to $\{0, 0.1, 0.1, ..., 1.9, 2\}$ and K belongs to $\{1, 2, ..., 19, 20\}$. Fig. 12(a) is a scatter plot showing 420 values of PSNR sorted from big to small, and the value of black dotted line corresponds to the PSNR value of the TV model for the same noisy image which is 26.304 dB (from Table 2). Fig. 12(b) is a hot map showing the value of PSNR with different σ and K. We can observe that the proposed algorithm E-a achieves better performance with most pairs of σ and K. The rate that PSNR value of FED-a is higher than the one of TV is about 87.38% (the number of PSNR higher than 26.304, 367 divided by the total number 420). In Fig. 12(c), we show the hot map of the number of iteration changing with σ and K, which indicate that the number of iteration decreases as σ and K increase.

In order to show the visual difference of different σ and K, several results of

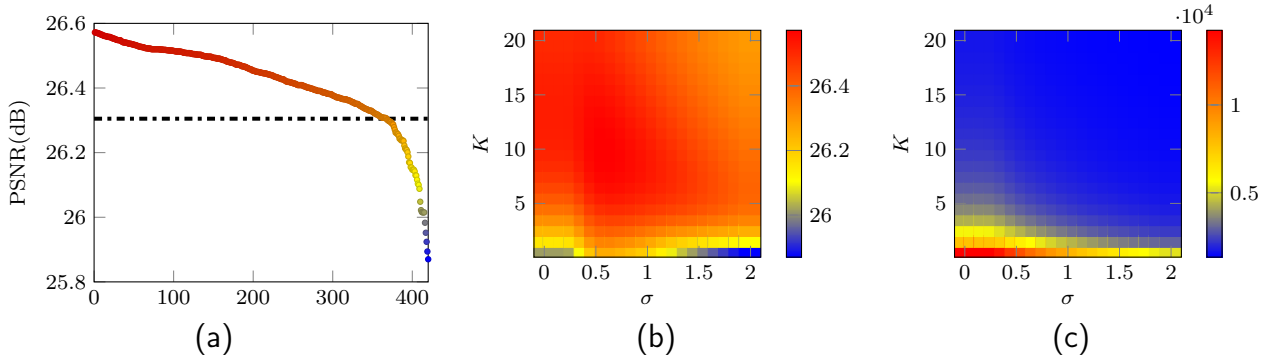

Figure 12: Robustness of parameters. (a) Scatter plot of PSNRs sorted form big to small from parametric experiments. (b) Hot map of values of PSNR with different σ and K. (c) Hot map of number of iterations with different σ and K.

the parametric experiment are depicted in Fig. 13. Fig. 13(f) - Fig. 13(h) are the denoising results of fixed σ, $\sigma = 6$, and these indicate that the smaller the K, the smoother the flat area is and the shaper the edge is in the large-scale sense. In other words, the restored image tends to be more cartoon image with small K. As illustrated in Fig. 3(b), the sensitive area of Φ_a changes with the difference of K. Specifically, Φ_a with $K = 1$ varies faster as $|\nabla G_\sigma * u| < 5$, and it is small and maintains a relatively gentle change as $|\nabla G_\sigma * u| > 5$. However, for $K = 20$, the change of Φ_a is very uniform in $|\nabla G_\sigma * u| < 60$ at least, and our model (3.3) is close to the LLT model as K tends to infinity. That is the reason why denoising results of our model look like more cartoon image when K is small, and look like the results of the LLT model when K is very large. In Fig. 13(i) - Fig. 13(l), we fix parameter $K = 9$ to investigate the influence of σ. From these images, we can see that a small σ tends to a result with clear edge and smooth flat. When σ is large, the result is not smooth enough in the place that should be flat area. The reason is that if σ is too large, $G * u$ would be too blurred which may destroy the edge information of the original image. In this case, adaptive coefficient does not fully utilize the feature information, causing the result tends to the resultant image of the LLT model. Therefore, we should avoid choosing K and σ too large, and suggest that $0.5 \leq \sigma \leq 1.5$ and $K \leq 16$.

5.3 Natural Images Denoising

For each natural image (Fig. 7(c) - Fig. 7(h)), we add Gaussian noise with σ_{n} belongs to $\{10, 20, 30, 40, 50\}$. Table 3 lists the average PSNR and SSIM results of our proposed fourth-order algorithms and competing methods on the six natural images. Without a doubt, BM3D achieve the best PSNR and SSIM results. Compared with PDE and variation based methods, our proposed algorithms perform better. As we can see, on average, PSNR of TV is higher than PSNRs of LLT, AFD, and MC, and SSIM of AFD is higher than SSIMs of TV, LLT, and MC. Our fourth-order algorithms outperform TV by 0.28dB to 0.46dB in PSNR, and outperform AFD by 0.01 to 0.02 in SSIM. The PSNR and SSIM results of different algorithms on the 6 test images are shown in Table 2. It can be seen that BM3D yields the highest PSNR and SSIM on most of images. However, for image "*piggy*", SSIMs of our proposed fourth-order algorithms is 0.03 to 0.1 higher than the SSIM of BM3D,

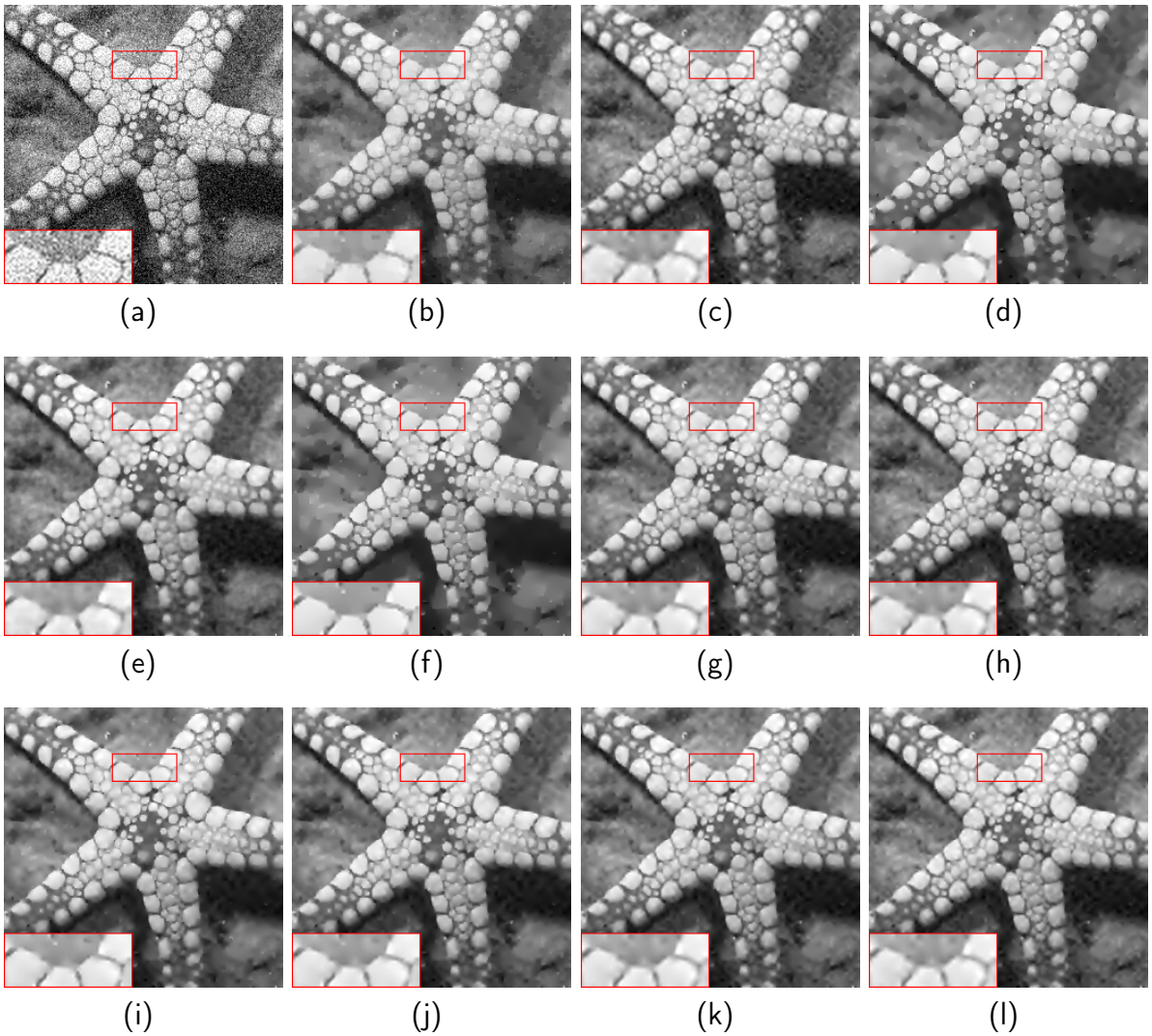

Figure 13: *starfish*(256×256), $\sigma_{\mathrm{n}} = 30$. (a) Noisy, $\sigma_{\mathrm{n}} = 30$. (b) TV. (c) LLT. (d) AFD. (e) MC. (f) E-a, $\sigma = 0.6$, $K = 1$. (g) E-a, $\sigma = 0.6$, $K = 15$. (h) E-a, $\sigma = 0.6$, $K = 20$. (i) E-a, $\sigma = 0$, $K = 9$. (j) E-a, $\sigma = 0.7$, $K = 9$. (k) E-a, $\sigma = 1.3$, $K = 9$. (l) E-a, $\sigma = 2$, $K = 9$.

which indicates the superiority in handling images with large smooth areas.

Fig. 14 - Fig.17 illustrate the visual results of different methods. As we can see that all of the methods can smooth noise well in a large range of noise level, and the TV model and our proposed models can preserve edges well. However, the staircase effects of the TV model is obvious, which derives false edges. The results of BM3D is very smooth with excellent visual effect. For the LLT model, the edges become blurred with the evolution progress. We can also observe that the LLT model and the MC model cannot recover flat regions well, and it leaves some 'picks' and 'valleys', causing a bad visual effect. All of the proposed fourth-order algorithms perform well for both of two coefficients. There are still some differences between these proposed models. For example, E-b could preserve more features than E-a, and the results of the AOS algorithm is smoother.

5.4 The Effectiveness of FED and AOS

The FED algorithm and the AOS algorithm can improve calculation efficiency and reduce the time spent, and these experiments of natural image denoising are all implemented in Matlab R2016b executed on a desktop PC whose processor is Intel(R) Core(TM) i7-7700K CPU @4.20GHz 4.20 GHz. The average CPU time is shown in Table 4. Specifically, because of the singularity of the fourth-order diffusion equation, the acceleration capability of the FED algorithm is limited, which is about 1.5 to 2 times faster on average. However, the implicit scheme can overcome some singularity of the proposed equation. The step size could be immense because of the unconditional stability, and the multiple of acceleration is about 3. At the same time, it should be noted that FED and AOS are very convenient to implement and do not increase the number of parameters.

6 Conclusion

Based on the LLT model, we have proposed an adaptive LLT model for image denoising. The experiments of 1D signal illustrate the superiority of adaptive model in preserving jumps. And then, from the perspective of the diffusion equation, the proposed adaptive LLT model has been generalized to a fourth-order diffusion framework. Different from other fourth-order models, the diffusion coefficients contain both first-order and second-order derivatives. Thus, the information of images is utilized fully. In the numerical implementation, we have designed an explicit scheme and a semi-implicit scheme, and employed the FED algorithm and the AOS algorithm to accelerate the efficiency, respectively. Finally, the experimental results have demonstrated the advantage of our models in edge preservation and flat area recovery.

References

[1] Leonid I. Rudin, Stanley Osher, and Emad Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D-nonlinear Phenomena*, 60(1-4):259–268, 1992.

Table 2: PSNR (dB) and SSIM results of denoising results of natural images by different methods.

σ_n	Images	TV	LLT	AFD	MC	BM3D	E-a	E-b	FED-a	FED-b	AOS-a	AOS-b
10	*piggy*	38.328	37.305	38.418	37.222	40.535	39.505	38.855	39.571	39.250	39.763	39.360
		0.614	0.555	0.629	0.505	0.627	0.657	0.652	0.649	0.645	0.670	0.668
	butterfly	32.716	32.194	32.231	32.473	33.257	33.158	33.067	33.074	32.964	33.143	33.0459
		0.812	0.795	0.812	0.797	0.830	0.821	0.821	0.818	0.817	0.823	0.8223
	castle	32.705	32.140	31.834	32.131	34.089	32.940	32.941	32.787	32.770	32.803	32.781
		0.563	0.554	0.548	0.550	0.626	0.561	0.562	0.559	0.554	0.562	0.565
	lena	32.926	33.061	32.535	33.019	34.513	33.370	33.345	33.230	33.203	33.386	33.336
		0.719	0.726	0.706	0.731	0.743	0.724	0.725	0.715	0.714	0.730	0.730
	starfish	32.224	32.368	31.651	32.190	33.305	32.805	32.757	32.720	32.672	32.684	32.605
		0.827	0.835	0.825	0.835	0.844	0.839	0.839	0.835	0.835	0.839	0.836
	parrot	32.362	31.787	31.592	31.766	33.365	32.505	32.467	32.376	32.308	32.434	32.378
		0.693	0.686	0.693	0.674	0.745	0.706	0.701	0.705	0.685	0.695	0.687
20	*piggy*	34.776	33.147	34.775	33.691	36.572	35.896	35.409	35.942	35.306	36.101	35.894
		0.541	0.460	0.560	0.432	0.531	0.606	0.595	0.595	0.587	0.617	0.626
	butterfly	28.628	27.698	28.486	28.647	29.575	29.033	28.964	29.054	28.942	29.196	29.041
		0.741	0.700	0.748	0.721	0.775	0.754	0.755	0.758	0.751	0.758	0.758
	castle	28.986	28.038	28.369	28.287	30.482	29.080	29.057	29.018	28.984	29.047	29.015
		0.437	0.420	0.440	0.431	0.506	0.452	0.441	0.443	0.435	0.446	0.441
	lena	29.567	29.582	29.416	29.592	31.314	29.990	29.944	29.941	29.895	30.070	30.043
		0.589	0.604	0.589	0.606	0.643	0.607	0.606	0.603	0.603	0.612	0.613
	starfish	28.342	28.307	28.065	28.344	29.655	28.777	28.713	28.740	28.678	28.782	28.713
		0.725	0.737	0.725	0.742	0.752	0.744	0.743	0.741	0.741	0.745	0.745
	parrot	28.725	27.863	28.240	28.257	29.749	28.906	28.826	28.864	28.768	28.937	28.876
		0.579	0.567	0.595	0.576	0.611	0.605	0.594	0.600	0.595	0.603	0.598
30	*piggy*	32.805	31.089	33.046	31.812	34.099	33.896	33.650	33.923	33.333	34.104	34.041
		0.498	0.427	0.528	0.405	0.490	0.553	0.546	0.554	0.540	0.574	0.582
	butterfly	26.345	25.334	26.424	26.591	27.648	26.856	26.732	26.826	26.713	27.017	26.793
		0.691	0.646	0.697	0.679	0.730	0.710	0.700	0.704	0.701	0.710	0.708
	castle	26.979	25.834	26.548	26.177	28.454	26.996	26.941	26.973	26.900	27.032	27.002
		0.376	0.351	0.390	0.367	0.441	0.399	0.390	0.414	0.387	0.417	0.397
	lena	27.841	27.685	27.735	27.740	29.489	28.189	28.115	28.171	28.073	28.183	28.269
		0.514	0.531	0.519	0.535	0.574	0.536	0.535	0.534	0.532	0.540	0.542
	starfish	26.305	26.111	26.152	26.149	27.525	26.563	26.506	26.540	26.479	26.622	26.554
		0.639	0.652	0.643	0.657	0.684	0.659	0.658	0.659	0.657	0.662	0.662
	parrot	26.677	25.633	26.432	26.271	27.616	26.876	26.774	26.858	26.746	26.947	26.848
		0.509	0.494	0.526	0.509	0.531	0.539	0.527	0.537	0.524	0.539	0.531
40	*piggy*	31.204	29.491	31.406	30.134	31.773	32.193	32.117	32.207	32.014	32.151	32.387
		0.466	0.413	0.506	0.367	0.452	0.527	0.516	0.522	0.513	0.523	0.560
	butterfly	24.837	23.764	24.995	25.077	26.097	25.396	25.318	25.423	25.302	25.435	25.408
		0.646	0.601	0.656	0.644	0.688	0.668	0.666	0.667	0.665	0.668	0.667
	castle	25.672	24.460	25.379	24.780	26.923	25.619	25.639	25.671	25.602	25.699	25.636
		0.325	0.299	0.341	0.316	0.380	0.343	0.356	0.360	0.367	0.368	0.367
	lena	26.657	26.381	26.528	26.479	28.139	26.938	26.863	26.925	26.844	26.975	26.977
		0.452	0.469	0.460	0.472	0.515	0.479	0.478	0.478	0.476	0.481	0.483
	starfish	24.962	24.734	24.785	24.697	25.907	25.189	25.143	25.171	25.124	25.237	25.151
		0.585	0.600	0.589	0.600	0.627	0.609	0.607	0.608	0.606	0.612	0.611
	parrot	25.318	24.286	25.270	24.966	25.904	25.640	25.528	25.622	25.507	25.739	25.613
		0.459	0.452	0.479	0.469	0.476	0.494	0.488	0.489	0.487	0.489	0.494
50	*piggy*	30.071	28.470	30.225	28.882	29.625	30.977	30.977	31.053	30.892	30.944	31.167
		0.446	0.399	0.472	0.342	0.439	0.490	0.496	0.492	0.491	0.497	0.524
	butterfly	23.581	22.625	23.872	23.926	24.465	24.189	24.151	24.169	24.135	24.258	24.277
		0.608	0.563	0.626	0.610	0.643	0.634	0.635	0.633	0.634	0.637	0.634
	castle	24.870	23.713	24.662	23.967	25.678	24.903	24.927	24.883	24.907	24.916	24.980
		0.285	0.262	0.302	0.276	0.330	0.313	0.322	0.308	0.316	0.307	0.334
	lena	25.720	25.308	25.505	25.389	26.866	25.873	25.779	25.864	25.745	25.847	25.888
		0.407	0.422	0.409	0.421	0.456	0.426	0.427	0.424	0.427	0.433	0.432
	starfish	23.966	23.781	23.783	23.779	24.348	24.144	24.110	24.128	24.094	24.191	24.122
		0.532	0.553	0.535	0.557	0.554	0.555	0.553	0.554	0.552	0.559	0.556
	parrot	24.302	23.222	24.385	23.931	24.401	24.629	24.527	24.639	24.513	24.715	24.643
		0.412	0.412	0.429	0.426	0.424	0.443	0.438	0.442	0.437	0.444	0.448

Table 3: Average PSNR (dB) and SSIM of 6 natural images

σ_n	TV	LLT	AFD	MC	BM3D	E-a	E-b	FED-a	FED-b	AOS-a	AOS-b
10	33.544	33.142	33.044	33.134	34.844	34.047	33.905	33.960	33.861	34.036	33.918
	0.705	0.692	0.702	0.682	0.736	0.718	0.717	0.714	0.708	0.720	0.718
20	29.837	29.106	29.559	29.470	31.224	30.280	30.152	30.260	30.096	30.355	30.264
	0.602	0.581	0.610	0.585	0.637	0.628	0.622	0.623	0.619	0.630	0.630
30	27.825	26.948	27.723	23.534	29.138	28.229	28.120	28.215	28.041	28.317	28.251
	0.538	0.517	0.551	0.450	0.575	0.566	0.559	0.567	0.557	0.574	0.570
40	26.442	25.519	26.394	26.022	27.457	26.829	26.768	26.836	26.732	26.873	26.862
	0.489	0.472	0.505	0.478	0.523	0.520	0.518	0.521	0.519	0.524	0.530
50	25.419	24.520	25.405	24.979	25.897	25.786	25.745	25.789	25.714	25.812	25.846
	0.448	0.435	0.462	0.439	0.474	0.477	0.478	0.475	0.476	0.479	0.488
Average	28.613	27.847	28.425	27.428	29.712	29.034	28.938	29.012	28.889	29.079	29.028
	0.556	0.539	0.566	0.527	0.589	0.582	0.579	0.580	0.576	0.585	0.587

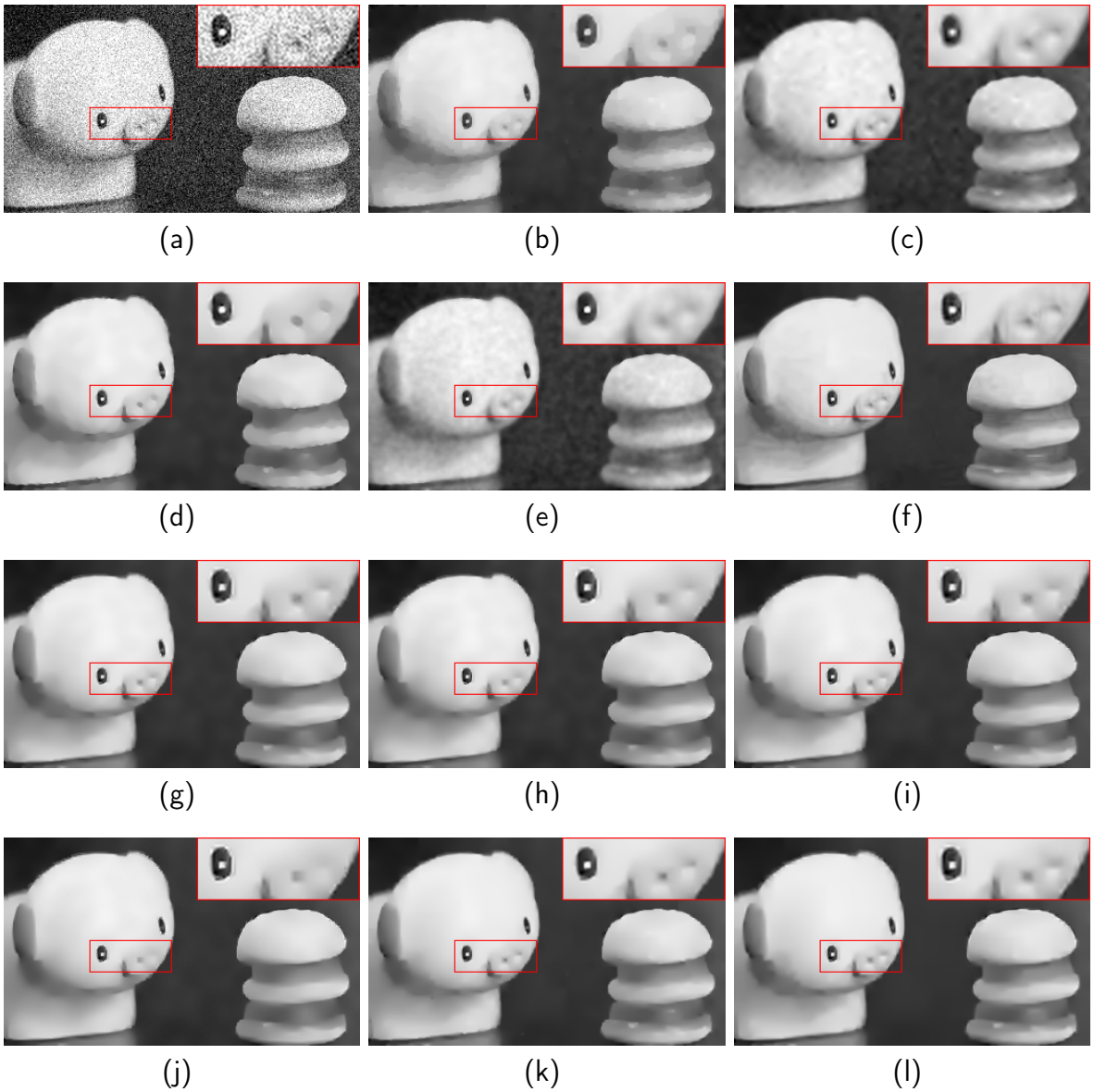

Figure 14: *piggy* (341 × 199), $\sigma_n = 30$. (a) Noisy, $\sigma_n = 30$. (b) TV. (c) LLT. (d) AFD. (e) MC. (f) BM3D. (g) E-a. (h) E-b. (i) FED-a. (j) FED-b. (k) AOS-a. (l) AOS-b.

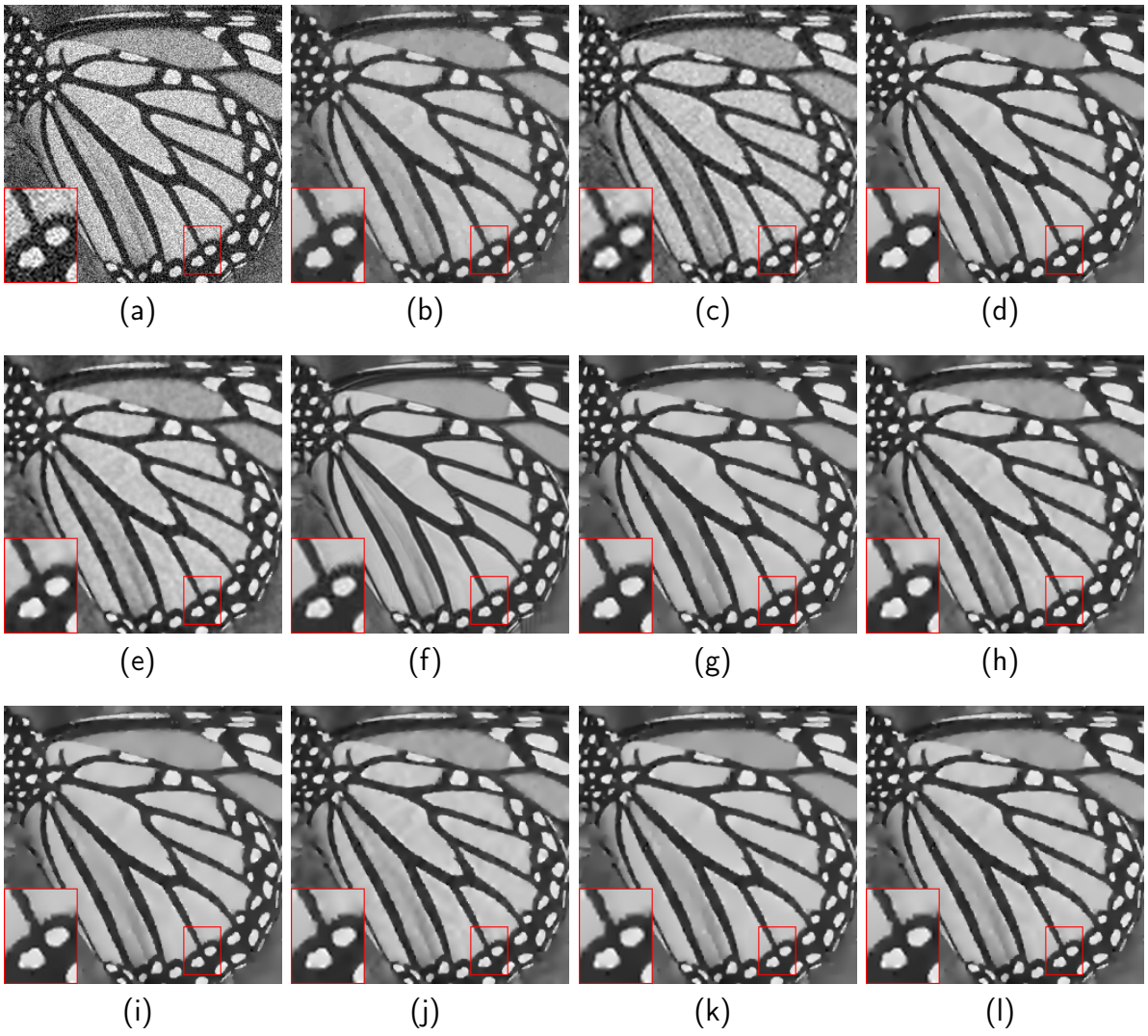

Figure 15: *butterfly* (256×256), $\sigma_{\mathrm{n}} = 30$. (a) Noisy, $\sigma_{\mathrm{n}} = 30$. (b) TV. (c) LLT. (d) AFD. (e) MC. (f) BM3D. (g) E-a. (h) E-b. (i) FED-a. (j) FED-b. (k) AOS-a. (l) AOS-b.

Table 4: Comparison of CPU time

σ_{n}	**E-b**	**FED-b**	**AOS-b**	**E-a**	**FED-a**	**AOS-a**
10	3.643	1.486	1.397	4.083	2.333	1.922
20	8.140	4.318	2.591	13.863	6.509	4.222
30	15.676	9.087	5.053	28.385	21.427	9.419
40	31.110	22.347	12.199	44.120	32.749	19.449
50	50.638	23.857	19.879	63.855	32.065	15.770

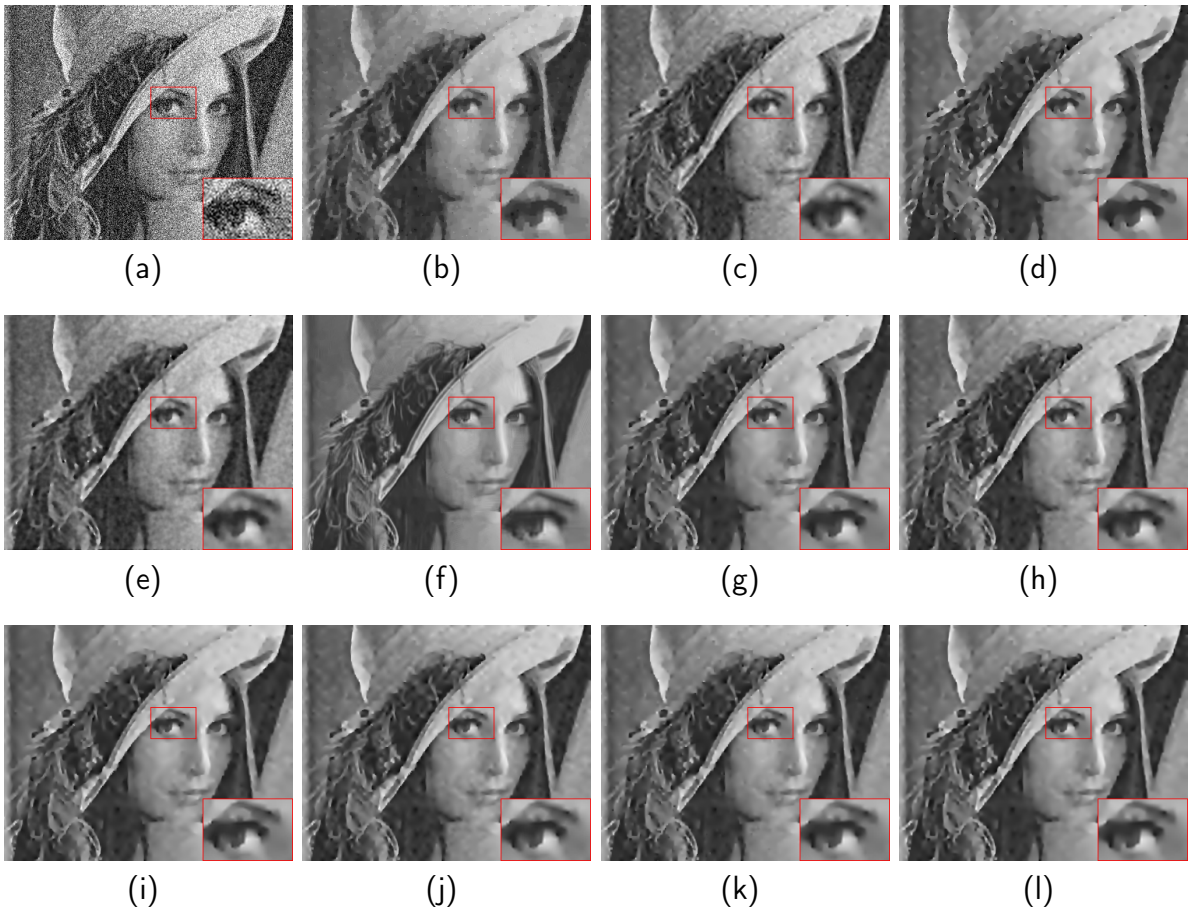

Figure 16: *lena* (371×302), $\sigma_{\mathrm{n}} = 40$. (a) Noisy, $\sigma_{\mathrm{n}} = 40$. (b) TV. (c) LLT. (d) AFD. (e) MC. (f) BM3D. (g) E-a. (h) E-b. (i) FED-a. (j) FED-b. (k) AOS-a. (l) AOS-b.

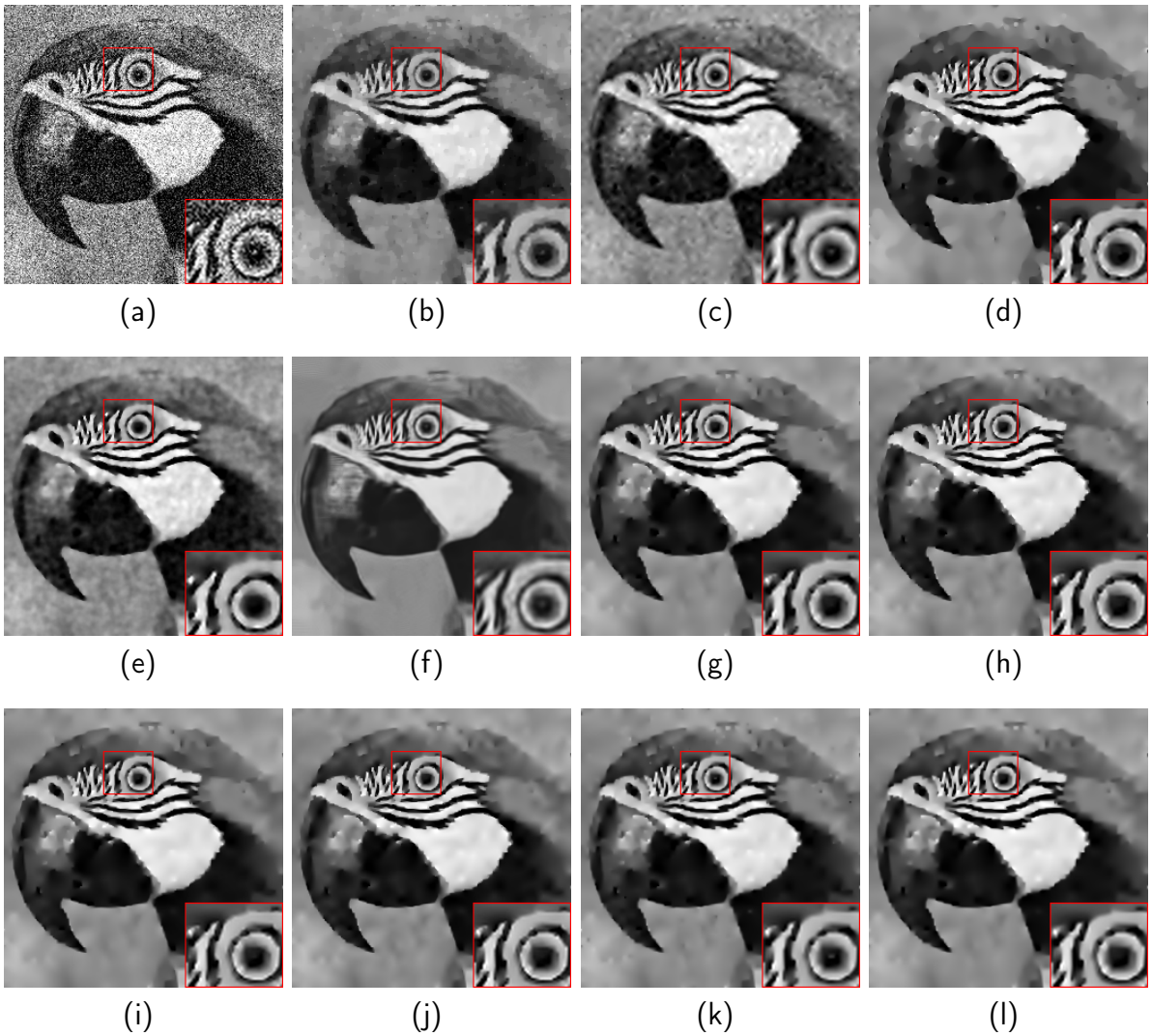

Figure 17: *parrot* (256×256), $\sigma_{\mathrm{n}} = 50$. (a) Noisy, $\sigma_{\mathrm{n}} = 50$. (b) TV. (c) LLT. (d) AFD. (e) MC. (f) BM3D. (g) E-a. (h) E-b. (i) FED-a. (j) FED-b. (k) AOS-a. (l) AOS-b.

[2] Marius Lysaker, Arvid Lundervold, and Xue-Cheng Tai. Noise removal using fourth-order partial differential equation with applications to medical magnetic resonance images in space and time. *IEEE Transactions on image processing*, 12(12):1579–1590, 2003.

[3] Marius Lysaker, Stanley Osher, and Xue-Cheng Tai. Noise removal using smoothed normals and surface fitting. *IEEE Transactions on Image Processing*, 13(10):1345–1357, 2004.

[4] Wei Zhu and Tony Chan. Image denoising using mean curvature of image surface. *SIAM Journal on Imaging Sciences*, 5(1):1–32, 2012.

[5] Carlos Brito-Loeza and Ke Chen. Multigrid algorithm for high order denoising. *SIAM Journal on Imaging Sciences*, 3(3):363–389, 2010.

[6] Francine Catté, Pierre-Louis Lions, Jean-Michel Morel, and Tomeu Coll. Image selective smoothing and edge detection by nonlinear diffusion. *SIAM Journal on Numerical analysis*, 29(1):182–193, 1992.

[7] Pietro Perona and Jitendra Malik. Scale-space and edge detection using anisotropic diffusion. *IEEE Transactions on pattern analysis and machine intelligence*, 12(7):629–639, 1990.

[8] Mohammad Reza Hajiaboli. An anisotropic fourth-order diffusion filter for image noise removal. *International Journal of Computer Vision*, 92(2):177–191, 2011.

[9] Tony Chan, Antonio Marquina, and Pep Mulet. High-order total variation-based image restoration. *SIAM Journal on Scientific Computing*, 22(2):503–516, 2000.

[10] Andrea Bertozzi and John Greer. Low-curvature image simplifiers: Global regularity of smooth solutions and laplacian limiting schemes. *Communications on Pure and Applied Mathematics: A Journal Issued by the Courant Institute of Mathematical Sciences*, 57(6):764–790, 2004.

[11] Andrea Bertozzi, John Greer, Stanley Osher, and Kevin Vixie. Nonlinear regularizations of tv based pdes for image processing. *Contemporary Mathematics*, 371:29–40, 2005.

[12] Seongjai Kim and Hyeona Lim. Fourth-order partial differential equations for effective image denoising. *Electron J Differ Equ*, 17:107–121, 2009.

[13] Marius Lysaker and Xue-Cheng Tai. Iterative image restoration combining total variation minimization and a second-order functional. *International journal of computer vision*, 66(1):5–18, 2006.

[14] Fang Li, Chaomin Shen, Jingsong Fan, and Chunli Shen. Image restoration combining a total variational filter and a fourth-order filter. *Journal of Visual Communication and Image Representation*, 18(4):322 – 330, 2007.

[15] Xinwu Liu, Lihong Huang, and Zhenyuan Guo. Adaptive fourth-order partial differential equation filter for image denoising. *Applied Mathematics Letters*, 24(8):1282 – 1288, 2011.

[16] Y-L You and Mostafa Kaveh. Fourth-order partial differential equations for noise removal. *IEEE Transactions on Image Processing*, 9(10):1723–1730, 2000.

[17] XY Liu, C-H Lai, and Kyriacos A Pericleous. A fourth-order partial differential equation denoising model with an adaptive relaxation method. *International Journal of Computer Mathematics*, 92(3):608–622, 2015.

[18] Mohammad Reza Hajiaboli. A self-governing hybrid model for noise removal. In *Pacific-Rim Symposium on Image and Video Technology*, pages 295–305. Springer, 2009.

[19] Peiying Chen and Yuandi Wang. Fourth-order partial differential equations for image inpainting. In *Audio, Language and Image Processing, 2008. ICALIP 2008. International Conference on*, pages 1713–1717. IEEE, 2008.

[20] Peiying Chen and Yuandi Wang. A new fourth-order equation model for image inpainting. In *Fuzzy Systems and Knowledge Discovery, 2009. FSKD'09. Sixth International Conference on*, volume 5, pages 320–324. IEEE, 2009.

[21] Peng Li, Shuai-Jie Li, Zheng-An Yao, and Zu-Jin Zhang. Two anisotropic fourth-order partial differential equations for image inpainting. *IET Image Processing*, 7(3):260–269, 2013.

[22] Shuaijie Li and Xiaohui Yang. Novel image inpainting algorithm based on adaptive fourth-order partial differential equation. *IET Image Processing*, 11(10):870–879, 2017.

[23] Bibo Lu and Qiang Liu. Image restoration with surface-based fourth-order partial differential equation. In *Visual Communications and Image Processing 2010*, volume 7744, page 774424. International Society for Optics and Photonics, 2010.

[24] Weili Zeng, Xiaobo Lu, and Xianghua Tan. A class of fourth-order telegraph-diffusion equations for image restoration. *Journal of Applied Mathematics*, 2011, 2011.

[25] Peng Li, Yang Zou, and Zhengan Yao. Fourth-order anisotropic diffusion equations for image zooming. *Journal of Image and Graphics*, 18(10):1261–1269, 2013.

[26] Konstantinos Papafitsoros and Carola-Bibiane Schönlieb. A combined first and second order variational approach for image reconstruction. *Journal of mathematical imaging and vision*, 48(2):308–338, 2014.

[27] Yadunath Pathak, KV Arya, and Shailendra Tiwari. Fourth-order partial differential equations based anisotropic diffusion model for low-dose ct images. *Modern Physics Letters B*, 32(25):1850300, 2018.

[28] Zhenyu Zhou, Zhichang Guo, Gang Dong, Jiebao Sun, Dazhi Zhang, and Boying Wu. A doubly degenerate diffusion model based on the gray level indicator for multiplicative noise removal. *IEEE Transactions on Image Processing*, 24(1):249–260, 2015.

[29] Kehan Shi, Dazhi Zhang, Zhichang Guo, and Boying Wu. A linear reaction-diffusion system with interior degeneration for color image compression. *SIAM Journal on Imaging Sciences*, 11(1):442–472, 2018.

[30] Sven Grewenig, Joachim Weickert, and Andrés Bruhn. From box filtering to fast explicit diffusion. In *Joint Pattern Recognition Symposium*, pages 533–542. Springer, 2010.

[31] Joachim Weickert, Sven Grewenig, Christopher Schroers, and Andrés Bruhn. Cyclic schemes for pde-based image analysis. *International Journal of Computer Vision*, 118(3):275–299, 2016.

[32] W Gentzsch and A Schlüter. Über ein einschrittverfahren mit zyklischer schrittweitenänderung zur lösung parabolischer differentialgleichungen. *Zeitschrift für angewandte Mathematik und Mechanik*, page T415, 1978.

[33] W Gentzsch. Numerical solution of linear and non-linear parabolic differential equations by a time-discretisation of third order accuracy. In *Proceedings of the third GAMM—Conference on Numerical Methods in Fluid Mechanics*, pages 109–117. Springer, 1980.

[34] Richard S Varga. *Matrix iterative analysis*, volume 27. Springer Science & Business Media, 2009.

[35] D Calvetti and L Reichel. Adaptive richardson iteration based on leja points. *Journal of computational and applied mathematics*, 71(2):267–286, 1996.

[36] Joachim Weickert, BM Ter Haar Romeny, and Max A Viergever. Efficient and reliable schemes for nonlinear diffusion filtering. *IEEE transactions on image processing*, 7(3):398–410, 1998.

[37] Joachim Weickert. Applications of nonlinear diffusion in image processing and computer vision. *Acta Math. Univ. Comenianae*, 70(1):33–50, 2001.

[38] T Lu, P Neittaanmaki, and X-C Tai. A parallel splitting-up method for partial differential equations and its applications to navier-stokes equations. *ESAIM: Mathematical Modelling and Numerical Analysis*, 26(6):673–708, 1992.

[39] Donald W Peaceman and Henry H Rachford, Jr. The numerical solution of parabolic and elliptic differential equations. *Journal of the Society for industrial and Applied Mathematics*, 3(1):28–41, 1955.

[40] Thomas P Witelski and Mark Bowen. Adi schemes for higher-order nonlinear diffusion equations. *Applied Numerical Mathematics*, 45(2-3):331–351, 2003.

[41] Willem Hundsdorfer and Jan G Verwer. *Numerical solution of time-dependent advection-diffusion-reaction equations*, volume 33. Springer Science & Business Media, 2013.

[42] Luca Calatroni, Bertram Düring, and Carola-Bibiane Schönlieb. Adi splitting schemes for a fourth-order nonlinear partial differential equation from image processing. *arXiv preprint arXiv:1305.5362*, 2013.

[43] Jim Douglas and James E Gunn. A general formulation of alternating direction methods. *Numerische Mathematik*, 6(1):428–453, 1964.

[44] Guri I Marchuk. Splitting and alternating direction methods. *Handbook of numerical analysis*, 1:197–462, 1990.

[45] Andrew Ronald Mitchell and David Francis Griffiths. The finite difference method in partial differential equations(book). *Chichester, Sussex, England and New York, Wiley-Interscience, 1980. 281 p*, 1980.

[46] Nikolaj Nikolaevič Janenko. *The method of fractional steps: the solution of problems of mathematical physics in several variables*. Springer, 1971.

[47] Kostadin Dabov, Alessandro Foi, Vladimir Katkovnik, and Karen Egiazarian. Image denoising by sparse 3-d transform-domain collaborative filtering. *IEEE Transactions on image processing*, 16(8):2080–2095, 2007.

[48] Sylvain Durand, Jalal Fadili, and Mila Nikolova. Multiplicative noise removal using l1 fidelity on frame coefficients. *Journal of Mathematical Imaging and Vision*, 36(3):201–226, 2010.

[49] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.