

## Article

# A Novel Reconstruction Algorithm with High Performance for Compressed Ultrafast Imaging

Qian Shen <sup>1,2</sup>, Jinshou Tian <sup>1</sup> and Chengquan Pei <sup>3,\*</sup>

<sup>1</sup> Key Laboratory of Ultra-Fast Photoelectric Diagnostics Technology, Xi'an Institute of Optics and Precision Mechanics, Xi'an 710049, China

<sup>2</sup> School of Optoelectronics, University of Chinese Academy of Sciences, Beijing 100049, China

<sup>3</sup> School of Artificial Intelligence, Xidian University, Xi'an 710071, China

\* Correspondence: peichengquan@xidian.edu.cn

**Abstract:** Compressed ultrafast photography (CUP) is a type of two-dimensional (2D) imaging technique to observe ultrafast processes. Intelligence reconstruction methods that influence the imaging quality are an essential part of a CUP system. However, existing reconstruction algorithms mostly rely on image priors and complex parameter spaces. Therefore, it usually takes a lot of time to obtain acceptable reconstruction results, which limits the practical application of the CUP. In this paper, we proposed a novel reconstruction algorithm named PnP-FFDNet, which can provide a high quality and high efficiency compared to previous methods. First, we built a forward model of the CUP and three sub-optimization problems were obtained using the alternating direction multiplier method (ADMM), and the closed-form solution of the first sub-optimization problem was derived. Secondly, inspired by the PnP-ADMM framework, we used an advanced denoising algorithm based on a neural network named FFDNet to solve the second sub-optimization problem. On the real CUP data, PSNR and SSIM are improved by an average of 3 dB and 0.06, respectively, compared with traditional algorithms. Both on the benchmark dataset and on the real CUP data, the proposed method reduces the running time by an average of about 96% over state-of-the-art algorithms, and show comparable visual results, but in a much shorter running time.

**Keywords:** compressed ultrafast photography; computational imaging; intelligent reconstruction algorithm



**Citation:** Shen, Q.; Tian, J.; Pei, C. A Novel Reconstruction Algorithm with High Performance for Compressed Ultrafast Imaging. *Sensors* **2022**, *22*, 7372. <https://doi.org/10.3390/s22197372>

Academic Editor: Lixiang Li

Received: 14 August 2022

Accepted: 26 September 2022

Published: 28 September 2022

**Publisher's Note:** MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



**Copyright:** © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

## 1. Introduction

Compressed ultrafast photography (CUP) is a new ultrafast computational imaging method, and it can achieve an imaging speed of  $10^{11}$  frames per second within a single shot. This technology introduces the framework of compressed sensing [1] into the imaging process of streak cameras, extending the imaging capability of streak cameras from one-dimensional (1D) to two-dimensional (2D). This emerging 2D ultrafast imaging technique is of great significance in revealing the fundamental mechanisms of physics, chemistry, and biomolecules [2]. It has a wide range of applications in the fields of fluorescence lifetime detection [3], real-time visualization of laser dynamics [4,5], and wide-field time-of-flight volume imaging [6], et al.

CUP reconstruction refers to establishing an inverse problem model based on the compressed sensing framework, and using an iterative algorithm to reconstruct 3D video data from the compressed 2D image captured by CUP. The reconstruction algorithm used in previous works in CUP is the two-step iterative threshold method (TwIST) [7]. Using total variation (TV) regularization in the TwIST reconstruction algorithm can easily induce artifacts [8], which limits the spatial-temporal resolution of CUP imaging and reconstruction efficiency. Ref. [9] introduced spatial and intensity constraints in the original TwIST algorithm by using an additional charge-coupled device (CCD) camera in the experimental

system to reduce low-intensity artifacts. Ref. [10] proposed applying the plug-and-play alternating direction method of multipliers (PnP-ADMM) [11] to the CUP reconstruction problem, and the block-matching 3D filtering denoising (BM3D) [12] algorithm is applied to the solution of the sub-problem of this algorithm, which improves the quality of the reconstruction and effectively suppresses the resolution anisotropy and artifacts. However, the variable separation strategy adopted does not take full advantage of the fact that the sensing matrix is a block-diagonal matrix, and the convergence of the PnP-ADMM algorithm applied to the CUP reconstruction problem is not explained. When applying the BM3D algorithm to video data, it is necessary to perform BM3D denoising for each frame, which is very time-consuming, while denoising algorithms based on neural networks have better denoising performance in computing speed.

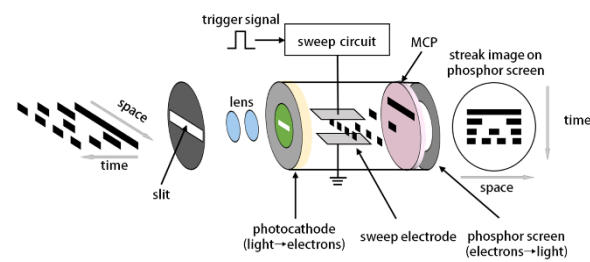
The alternating direction multiplier method (ADMM) is a widely used algorithm for constrained optimization problems in image restoration. Based on ADMM, the PnP-ADMM algorithm implements a modular structure. The biggest advantage of PnP-ADMM is that it allows the state-of-the-art denoising algorithms to be applied to the solution process of sub-optimization problems without specifying specific priors, which greatly improves the flexibility of the algorithm. Therefore, an excellent denoising algorithm can be found and inserted into the algorithm framework of PnP-ADMM to improve the reconstruction quality of the algorithm. Existing image denoising algorithms can be divided into two categories: model-based methods and discriminative-learning-based methods. Similar to total variation denoising (TVD) [13], BM3D, and weighted nuclear norm minimization (WNNM) [14] for image denoising, these algorithms are flexible in dealing with denoising problems with different noise levels, but they have some drawbacks. For example, algorithms are generally time-consuming and have many parameters that need to be manually tuned. Furthermore, these algorithms usually rely on manually determined priors, such as sparsity [15,16] and non-local self-similarity [17–19], which have limitations for describing complex image structures. Discriminant learning has been widely studied in image denoising due to its advantages of fast inference speed and good performance. Some examples include learning deep CNN denoiser prior for image restoration (IRCnn) [20], deep CNN for image denoising (DnCNN) [21], and fast and flexible solution for CNN-based image denoising (FFDNet) [22]. Their non-linear mapping layer is a collection of “Convolution + Batch Normalization + Rectified Linear Units” layers with filters of spatial size  $3 \times 3$ . Among them, FFDNet has several desirable properties that make it very suitable as a denoiser to be applied to the framework of PnP-FFDNet. First, it introduces a noise map as an input channel, so that a single model can handle a wide range of noise levels, so that it exhibits apparent results on both synthetic noisy images corrupted by additive white Gaussian noise (AWGN) and real-world noisy images [22,23]. Second, FFDNet reduces the size of the input by down-sampling, which makes it faster for forward inference. Therefore, compared with the model-based algorithm BM3D, with excellent de-noising performance even on CPU, FFDNet is faster without sacrificing the denoising performance. These properties make FFDNet very suitable as a denoiser in the framework of PnP-ADMM. In this paper, we proposed a novel reconstruction algorithm based on PnP and FFDNet to reconstruct the CUP system. FFDNet can learn the noise model in CUP system well. The performance on both simulated datasets and real data shows that our method performs well on both reconstructed visual effects and metric evaluation, and the reconstruction time is greatly reduced.

## 2. Forward Model of CUP

### 2.1. Principle of Streak Camera

Streak camera is an ultrafast imaging device that can capture dynamic events that occur on picosecond or even femtosecond timescales. As shown in Figure 1, a long slit with width of several microns is usually inserted in front of streak camera. The optical signal is converted into an electrical signal through the streak image tube and space-time mapping is performed by an ultrafast scanning unit. Microchannel plate (MCP) realizes

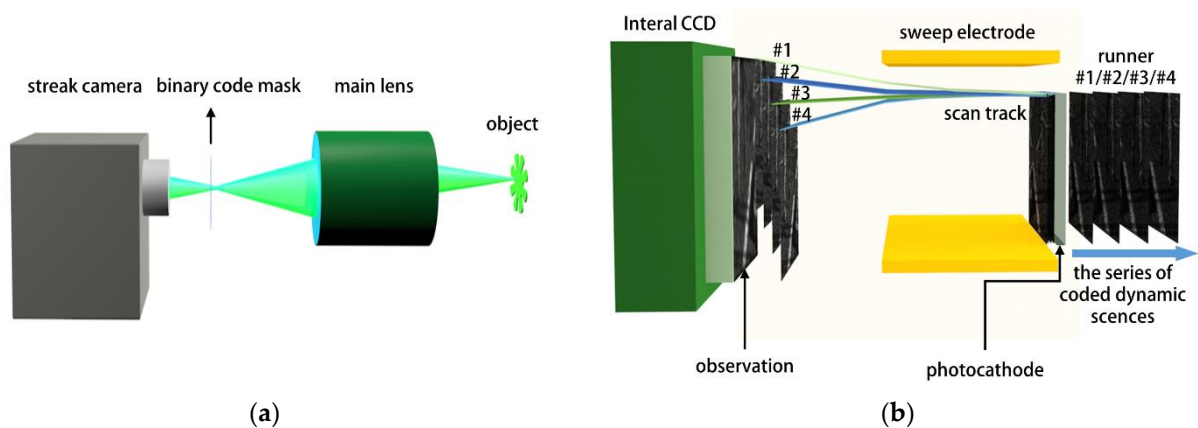
the multiplication of photoelectrons, and the phosphor screen converts the photoelectrons into optical signals.



**Figure 1.** Schematic diagram of the imaging process of the streak camera in one-dimensional view.

## 2.2. Design of Compressed Ultrafast Photography

Figure 2a shows a schematic diagram of the CUP imaging system. The experimental system is mainly composed of streak camera (integrated CCD), a random binary mask, and a main camera lens. The random binary mask performs intensity modulation on the image of the detection target passing through the main lens. The light intensity modulation factor of the transparent area of the mask is 1, and the light intensity modulation factor of the opaque area of the mask is 0. The difference between CUP and the traditional streak camera is that the CUP requires the slit on the streak camera to be fully opened. This is the key to CUP's ability to perform 2D ultrafast imaging. Figure 2b shows how the streak camera works in the CUP. Since there is no limitation of the slit, the one to four frames of the encoded images entered into the streak camera and were scanned by an ultrafast electric field. Then, the second frame was sheared by a pixel compared with the first frame in the scanning axis, and this was also performed on the third, fourth frames, etc. Finally, all the images were accumulated into a compressed image and recorded by a CCD camera. Next, the intelligence methods were performed to reconstruct the dynamic ultrafast video from a compressed image.



**Figure 2.** Schematic diagram of the CUP system. (a) Schematic diagram of the optical path structure of the CUP. (b) Schematic diagram of the time series of the encoded dynamic scene being shifted after entering the streak camera.

Figure 3 describes the basic workflow of the CUP system. Four frames are selected from the benchmark dataset runner to simulate the dynamic scene  $I(x, y, t)$ , with different frames simulating the moment when the dynamic scene occurs. Each frame is of size  $256 \times 256$ , and then encoded by random matrix, which is a  $\{0, 1\}$  randomly distributed binary code. The symbol  $\odot$  denotes the Hadamard (element-wise) product. The data compression of CUP follows these three steps:

Step 1: Encoding. Each frame of the dynamic scene  $I(x, y, t)$  is  $\odot$  with the same mask, and the encoding operation is denoted as  $C$ . The encoded dynamic scene is  $CI(x, y, t)$ ;

Step 2: Shift. A deflection electrode inside the streak camera provides a deflection voltage in the vertical direction, so the frames that arrived at different times were shifted in the vertical direction at different positions. The direction of the offset of each frame is marked in Figure 3. For the convenience of mathematical simplification later, it is assumed that each frame is offset by  $s_0$  pixels, the shift operation is recorded as  $S$ , and the dynamic coding scene after translation is  $SCI(x, y, t)$ ;

Step 3: Overlay. The receiver of the streak camera is an internal CCD. During the exposure time of the CCD, photons arriving at the CCD at different times are accumulated. The  $SCI(x, y, t)$  in step 2 is superimposed along the time axis, denoted as  $T$ , and the two-dimensional observation result  $Y(m, n) = TSCI(x, y, t)$  is finally obtained.

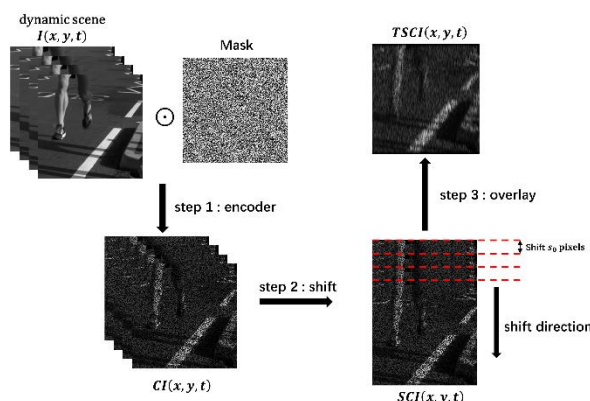


Figure 3. Data compression flowchart for CUP.

The goal of CUP reconstruction is to reconstruct the original three-dimensional dynamic scene  $I(x, y, t)$  from the obtained two-dimensional observation  $Y(m, n)$ . Without loss of generality, the dynamic scene can be viewed as video data with  $N$  frames, and will mathematically describe the CUP imaging process and build a classical inverse problem model.

The video data  $X(X \in \mathbb{R}^{n_x \times n_y \times N})$  that are the matrix representation of the dynamic scene  $I(x, y, t)$  with  $N$  frames are compressed into one frame of two-dimensional observation data  $Y(Y \in \mathbb{R}^{L \times n_y})$ , which are the matrix representation of the observation  $Y(m, n)$  by the CUP system. Figure 4 shows the relationship between  $L$  and  $n_x$ . Therefore,  $L = (N - 1)s_0 + n_x$ . The coding matrix  $C(C \in \mathbb{R}^{n_x \times n_y})$  used by the CUP system is random binary code. The data compression observation model of CUP [10] can be expressed as:

$$Y = TSCX + Z \tag{1}$$

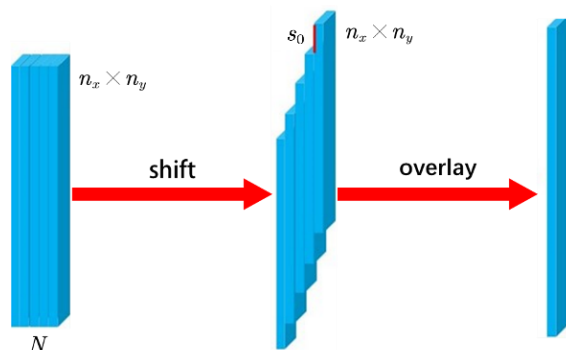
where  $Z(Z \in \mathbb{R}^{L \times n_y})$  represents the noise in the observation process,  $X(X \in \mathbb{R}^{n_x \times n_y \times N})$  can be expressed as  $[X_1, X_2, \dots, X_N]^T$ ,  $C$  represents the encoding process,  $S$  represents the shift process, and  $T$  represents the accumulating process.  $T$  can be represented by the summation notation:

$$Y = \sum_{i=1}^N S_i(C \odot X_i) + Z \tag{2}$$

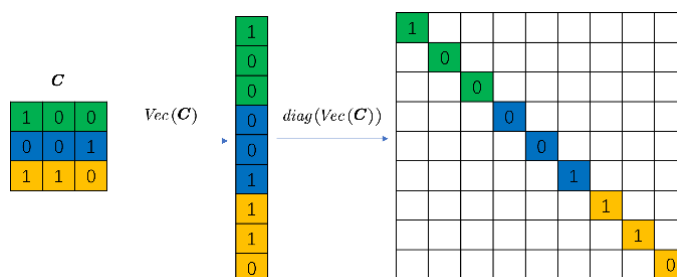
where  $\odot$  denotes the Hadamard (element-wise) product. To convert the Hadamard product to matrix multiplication,  $Y(Y \in \mathbb{R}^{L \times n_y})$ ,  $X(X \in \mathbb{R}^{n_x \times n_y \times N})$ , and  $Z(Z \in \mathbb{R}^{L \times n_y})$  are transformed into 1D column vectors:

$$\begin{aligned} y &= \sum_{i=1}^N S'_i(C_{diag}x_i) + z \\ &= \sum_{i=1}^N (S'_i C_{diag})x_i + z \\ &= [S'_1 C_{diag}, S'_2 C_{diag}, \dots, S'_N C_{diag}] [x_1^T, x_2^T, \dots, x_N^T]^T + z \end{aligned} \tag{3}$$

where  $x_i = \text{Vec}(\mathbf{X}_i) \in \mathbb{R}^{n_x n_y \times 1}$ ,  $y = \text{Vec}(\mathbf{Y}) \in \mathbb{R}^{L n_y \times 1}$  and  $z = \text{Vec}(\mathbf{Z}) \in \mathbb{R}^{L n_y \times 1}$ ,  $\mathbf{C}_{diag} = \text{diag}(\text{Vec}(\mathbf{C})) \in \mathbb{R}^{n_x n_y \times n_x n_y}$ . Figure 5 shows a simple example when the mask is  $3 \times 3$ .



**Figure 4.** For a data cube of size  $n_x \times n_y \times N$ . Its height is  $n_x$ , width is  $n_y$ , and the number of frames is  $N$ . Then, the compressed image size is  $[(N - 1)s_0 + n_x] \times n_y$ .



**Figure 5.** Example when mask is a  $3 \times 3$  matrix. Different colours represent different rows of  $\mathbf{C}$ .

Equation (3) can be mathematically described as a classic inverse problem model as follows:

$$y = Hx + z \tag{4}$$

where  $x = [x_1^T, x_2^T, \dots, x_N^T]^T$ , and the sensing matrix  $H \in \mathbb{R}^{L n_y \times n_x n_y N}$  is a block-diagonal matrix,  $H$  can be expressed as:

$$H = [H_1, \dots, H_N] = [S'_1 C_{diag}, S'_2 C_{diag}, \dots, S'_N C_{diag}] \tag{5}$$

where  $S'_i = \text{CircShift}(I_0, (i - 1)s_0 n_y) \in \mathbb{R}^{L n_y \times n_x n_y}$ ,  $i = 1, 2, 3, \dots, N$ ,  $s_0$  indicates the number of pixels per shift.  $\text{CircShift}(A, l)$  represents the circular translation of  $l$  pixels along the vertical direction of the matrix  $A$ .  $I_0 \in \mathbb{R}^{L n_y \times n_x n_y}$  can be expressed as:

$$I_0 = \begin{bmatrix} I_{n_x n_y \times n_x n_y} \\ \mathbf{O}_{(N-1)s_0 n_y \times n_x n_y} \end{bmatrix} \tag{6}$$

where  $I \in \mathbb{R}^{n_x n_y \times n_x n_y}$  represents the identity matrix. A simple case of  $H_i$  in Equation (5) when the number of frames is four is shown in Figure 6, and it can be written as:  $H_i = \text{CircShift}(C_0, (i - 1)s_0 n_y) \in \mathbb{R}^{L n_y \times n_x n_y}$ ,  $i = 1, 2, 3, \dots, N$  and  $C_0 \in \mathbb{R}^{L n_y \times n_x n_y}$  can be expressed as:

$$C_0 = \begin{bmatrix} (C_{diag})_{n_x n_y \times n_x n_y} \\ \mathbf{O}_{(N-1)s_0 n_y \times n_x n_y} \end{bmatrix} \tag{7}$$

An important property of the sensing matrix is that  $\mathbf{H}\mathbf{H}^T$  is a diagonal matrix, which is used in Section 3.1 to obtain the closed-form solution of the suboptimization problem.  $\mathbf{H}\mathbf{H}^T$  can be written as  $\mathbf{H}\mathbf{H}^T = \sum_{i=1}^N \mathbf{H}_i\mathbf{H}_i^T$  and consider that when  $i = 1$ , the result of  $\mathbf{H}_1\mathbf{H}_1^T$  is:

$$\mathbf{H}_1\mathbf{H}_1^T = \begin{bmatrix} \mathbf{C}^{diag} \\ \mathbf{O} \end{bmatrix} \begin{bmatrix} \mathbf{C}^{diag} & \mathbf{O} \end{bmatrix} = \begin{bmatrix} \mathbf{C}^{diag} & \mathbf{O} \\ \mathbf{O} & \mathbf{O} \end{bmatrix} \tag{8}$$

where  $\mathbf{C}^{diag} = \text{diag}(\text{Vec}(\mathbf{C}))_{n_x n_y \times n_x n_y}$  is a diagonal matrix and  $\mathbf{O}$  represent the zero matrix. In the case of  $i = 1$ , the result of  $\mathbf{H}_1\mathbf{H}_1^T$  is a diagonal matrix. When generalized to  $i = 2, \dots, N$ , the result of  $\mathbf{H}_i\mathbf{H}_i^T$  is still a diagonal matrix, so their sum is also naturally a diagonal matrix and, thus,  $\mathbf{H}\mathbf{H}^T$  is a diagonal matrix.

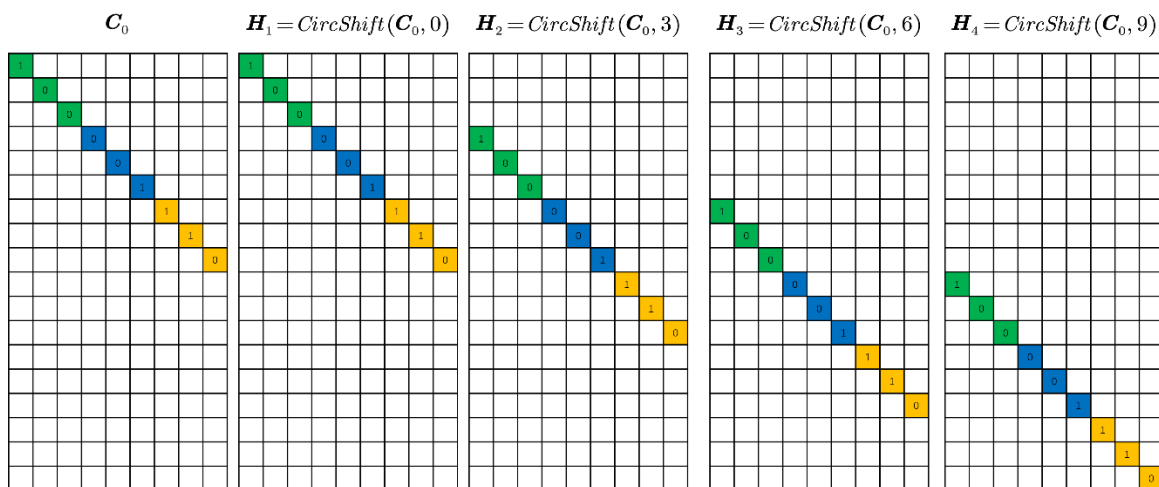


Figure 6. Example of the matrix  $\mathbf{H}$ . Different colours represent different rows of  $\mathbf{C}$ .

### 3. Novel Reconstruction Method for CUP

#### 3.1. Algorithm Framework of PnP-ADMM for CUP

According to the forward model of CUP data compression already established in Section 2.2, the inverse problem model of Equation (4) can be described as an unconstrained optimization problem:

$$\hat{\mathbf{x}} = \underset{\mathbf{x}}{\text{argmin}} f(\mathbf{x}) + \lambda g(\mathbf{x}) \tag{9}$$

In the formula,  $f(\mathbf{x}) = \|\mathbf{y} - \mathbf{H}\mathbf{x}\|_2^2$  represents the CUP forward imaging model, and  $g(\mathbf{x})$  represents a certain image prior.

The ADMM transforms the unconstrained optimization problem (9) into a constrained optimization problem by introducing an auxiliary variables  $\mathbf{v}$ :

$$\underset{\mathbf{x}, \mathbf{v}}{\text{argmin}} \left( \hat{\mathbf{x}}, \hat{\mathbf{v}} \right) = \underset{\mathbf{x}, \mathbf{v}}{\text{argmin}} f(\mathbf{x}) + \lambda g(\mathbf{v}), \text{subject to } \mathbf{x} = \mathbf{v} \tag{10}$$

The minimum optimization problem (10) can be solved by iteratively solving the following three sub-optimization problems [24]:

$$\mathbf{x}^{(k+1)} = \underset{\mathbf{x}}{\text{argmin}} f(\mathbf{x}) + \frac{\rho}{2} \|\mathbf{x} - \left( \mathbf{v}^{(k)} - \frac{1}{\rho} \mathbf{u}^{(k)} \right)\|_2^2 \tag{11}$$

$$\mathbf{v}^{(k+1)} = \underset{\mathbf{v}}{\text{argmin}} \lambda g(\mathbf{v}) + \frac{\rho}{2} \|\mathbf{v} - \left( \mathbf{x}^{(k)} + \frac{1}{\rho} \mathbf{u}^{(k)} \right)\|_2^2 \tag{12}$$

$$\mathbf{u}^{(k+1)} = \mathbf{u}^{(k)} + \rho \left( \mathbf{x}^{(k+1)} - \mathbf{v}^{(k+1)} \right) \tag{13}$$

where  $k$  represents the number of iterations.

In the CUP reconstruction problem,  $f(x) = \frac{1}{2}\|y - Hx\|_2^2$  represents the forward model of CUP imaging. For the convenience of representation, the sub-problem (11) is rewritten in the form without the iteration variable  $k$ :

$$x = \operatorname{argmin}_x \frac{1}{2}\|y - Hx\|_2^2 + \frac{\rho}{2}\|x - \left(v - \frac{1}{\rho}u\right)\|_2^2 \quad (14)$$

For the determined  $v, u, H, y$ , sub-problem (14) has a closed-form solution:

$$x = \left(H^T H + \rho I\right)^{-1} \left[ H^T y + \rho \left(v - \frac{1}{\rho}u\right) \right] \quad (15)$$

For a large matrix  $H$ , inverting  $(H^T H + \rho I)$  will consume a lot of computer memory resources and time. For the original data  $X$  with size of  $256 \times 256 \times 8$ , the size of  $H$  is  $67,328 \times 524,288$ , and the size of  $H^T H$  is  $524,288 \times 524,288$ . At present, it is difficult for computers to deal with such a large-scale matrix inversion problem. Inspired by [25], when  $HH^T$  is a diagonal matrix, Equation (14) has a closed-form solution. It was discussed in Section 2.2 that the matrix  $H$  is a block-diagonal matrix, and it is verified that  $HH^T$  is a diagonal matrix. This feature of  $H$  can simplify the computation of the inversion of  $(H^T H + \rho I)$ :

$$\left(H^T H + \rho I\right)^{-1} = \rho^{-1}I - \rho^{-1}H^T \left(I + \rho^{-1}HH^T\right)^{-1} H\rho^{-1} \quad (16)$$

Bringing Equation (16) into Equation (15), we can obtain [25]:

$$x = \rho^{-1} \left[ H^T y + \rho \left(v - \frac{1}{\rho}u\right) \right] - \rho^{-2}H^T \left(I + \rho^{-1}HH^T\right)^{-1} HH^T y - \rho^{-1}H^T \left(I + \rho^{-1}HH^T\right)^{-1} H \left(v - \frac{1}{\rho}u\right) \quad (17)$$

where  $HH^T$  is a diagonal matrix:

$$HH^T = \operatorname{diag}\left(\left[\psi_1, \dots, \psi_{L_{Ny}}\right]\right) \quad (18)$$

then we can obtain:

$$\left(I + \rho^{-1}HH^T\right)^{-1} = \operatorname{diag}\left(\left[\frac{\rho}{\rho + \psi_1}, \dots, \frac{\rho}{\rho + \psi_{L_{Ny}}}\right]\right) \quad (19)$$

$$\left(I + \rho^{-1}HH^T\right)^{-1} HH^T = \operatorname{diag}\left(\left[\frac{\rho\psi_1}{\rho + \psi_1}, \dots, \frac{\rho\psi_{L_{Ny}}}{\rho + \psi_{L_{Ny}}}\right]\right) \quad (20)$$

If  $y = [y_1, \dots, y_{L_{Ny}}]^T$ ,  $[H(v - \frac{1}{\rho}u)]_i$  denote the  $i$ th element of vector  $[H(v - \frac{1}{\rho}u)]$ , Equation (17) becomes [25]:

$$x = \rho^{-1}H^T y + \left(v - \frac{1}{\rho}u\right) - \rho^{-1}H^T \left[ \frac{y_1\psi_1 + \rho[H(v - \frac{1}{\rho}u)]_1}{\rho + \psi_1}, \dots, \frac{y_{L_{Ny}}\psi_{L_{Ny}} + \rho[H(v - \frac{1}{\rho}u)]_{L_{Ny}}}{\rho + \psi_{L_{Ny}}} \right]^T \left(v - \frac{1}{\rho}u\right) + H^T \left[ \frac{y_1 - [H(v - \frac{1}{\rho}u)]_1}{\rho + \psi_1}, \dots, \frac{y_{L_{Ny}} - [H(v - \frac{1}{\rho}u)]_{L_{Ny}}}{\rho + \psi_{L_{Ny}}} \right]^T \quad (21)$$

Using matrix division of corresponding elements, Equation (21) can be simplified to:

$$\mathbf{x} = \left( \mathbf{v} - \frac{1}{\rho} \mathbf{u} \right) + \mathbf{H}^T \left[ \mathbf{y} - \mathbf{H} \left( \mathbf{v} - \frac{1}{\rho} \mathbf{u} \right) \right] ./ (\mathbf{s} + \rho \mathbf{I}) \quad (22)$$

where  $\mathbf{s} = [\psi_1, \dots, \psi_{L_{ny}}]^T$ , in Equation (22), the matrix division of the corresponding element has priority.

By utilizing the property that  $\mathbf{H}\mathbf{H}^T$  is a diagonal matrix, in each iteration process, the solution of sub-optimization problem (11) can be completed with only one calculation, so the computer memory load is reduced and the solution efficiency of the algorithm is improved.

One of the most important features of ADMM iteration is its modular structure: problem (11) can be seen as a reversal step, since it includes the forward imaging model  $f(\mathbf{x})$ , and problem (12) can be seen as a denoising step, because it includes the image a priori  $g(\mathbf{v})$ . If  $\sigma = \sqrt{\frac{\lambda}{\rho}}$ , problem (12) can be rewritten as [24]:

$$\mathbf{v}^{(k+1)} = \underset{\mathbf{v}}{\operatorname{argmin}} \lambda g(\mathbf{v}) + \frac{1}{2\sigma^2} \left\| \mathbf{v} - \left( \mathbf{x}^{(k)} + \frac{1}{\rho} \mathbf{u}^{(k)} \right) \right\|_2^2 \quad (23)$$

Considering  $\left( \mathbf{x}^{(k)} + \frac{1}{\rho} \mathbf{u}^{(k)} \right)$  as a noisy image, problem (23) minimizes the two-norm distance between the noise-free image  $\mathbf{v}$  and the noisy image  $\left( \mathbf{x}^{(k)} + \frac{1}{\rho} \mathbf{u}^{(k)} \right)$ , based on the image prior  $g(\mathbf{v})$  as a regular term. If  $g(\mathbf{v}) = \|\mathbf{v}\|_{TV}$ , where  $\|\cdot\|_{TV}$  represents the total variation norm, it can be calculated by Equation (24) [24]:

$$\|\mathbf{v}\|_{TV} = \sum_i \sqrt{(\Delta_i^h, \mathbf{v})^2 + (\Delta_i^v, \mathbf{v})^2} \quad (24)$$

Then problem (23) becomes the standard total variation norm denoising problem, namely the TV denoising problem. Formally based on this intuition, [11] proposed the PnP-ADMM method, without specifying the image prior  $g(\mathbf{v})$ , and just replaces step (12) with a state-of-the-art image denoising algorithm [24]:

$$\mathbf{v}^{(k+1)} = \mathcal{D}_\sigma \left( \mathbf{x}^{(k)} + \frac{1}{\rho} \mathbf{u}^{(k)} \right) \quad (25)$$

where  $\mathcal{D}_\sigma(\cdot)$  represents some type of noise denoising algorithm. Although it is unclear to which image prior the denoising algorithm  $\mathcal{D}_\sigma(\cdot)$  corresponds, the performance of the PnP-ADMM method on the image reconstruction problem surpasses other popular image reconstruction algorithms [26–29].

### 3.2. The Architecture of FFDNet

Proposed by Zhang et al. in [22], FFDNet is a single discriminative CNN model. Figure 7 shows the architecture of FFDNet. FFDNet consists of “Downsampling layer + Nonlinear mapping layer + Upsampling layer”. The down-sampling layer is a reversible down-sampling operator that reshapes a noisy image into four down-sampled sub-images. At the same time, FFDNet concatenates a tunable noise map with the down-sampled sub-images to form a tensor as the inputs to the non-linear mapping layer. At the non-linear mapping layer, each sub-layer is composed of a specific combination of three types of operations: convolution (Conv) with filter size of  $3 \times 3$ , rectified linear units (ReLU), and batch normalization (BN). For the grayscale model, the number of Conv layer is 15 and the number of channels is 64. The noise map varies from 0 to 75 [23]. After the non-linear mapping layer, an upscaling operation is applied in the up-sampling layer as the reverse operator of the down-sampling operator applied in the input stage to produce the estimated clean image with the same shape as the input noisy image.



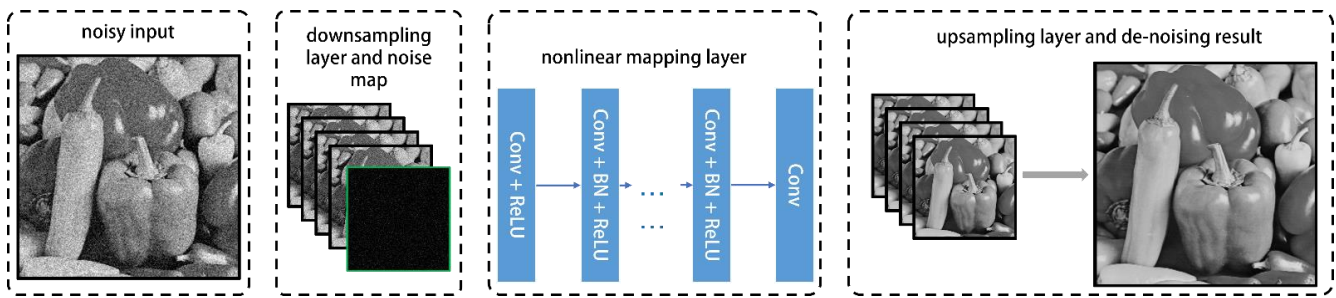


Figure 7. Architecture of FFDNet [22].

The training dataset is composed of pairs of input-output patches  $\left\{ \left( \left( \tilde{I}_j, M_j \right), I_j \right) \right\}_{j=0}^m$ , which are generated by adding AWGN of  $\sigma \in [0, 75]$  to clean patches  $I_j$  and build the corresponding noise map  $M_j$ . FFDNet, without a residual learning estimate, can denoise the image directly [23]:

$$\mathcal{F}(\tilde{I}) = \hat{I} \quad (26)$$

thus, the corresponding loss function is [23]:

$$\mathcal{L}(\theta) = \frac{1}{2m} \sum_{j=1}^m \left\| \mathcal{F}(\tilde{I}_j, M_j; \theta) - I_j \right\|^2 \quad (27)$$

where  $\theta$  is the collection of all learnable parameters. Therefore, the architecture and these additional techniques render this algorithm faster, more efficient, and more versatile than other denoising algorithms.

### 3.3. PnP-ADMM Fixed-Point Convergence for CUP Reconstruction

Ref. [24] demonstrates the fixed-point convergence of the PnP-ADMM algorithm based on the definition of a bounded denoiser and the assumption of bounded gradients.

**Definite 1.** Bounded denoiser: A bounded denoiser with a parameter  $\sigma$  is a function  $\mathcal{D}_\sigma : \mathbb{R}^n \rightarrow \mathbb{R}^n$  such that for any input  $x \in \mathbb{R}^n$  [24]:

$$\frac{1}{n} \left\| \mathcal{D}_\sigma(x) - x \right\|_2^2 \leq \sigma^2 C \quad (28)$$

for some universal constant  $C$ , independent of  $n$  and  $\sigma$ .

Bounded denoisers are a weak condition that we expect most denoisers to have. Next, we show that bounded gradients also hold in the CUP reconstruction problem. In the problem of CUP reconstruction, the gradient of  $f(x)$  is:

$$\nabla f(x) = \mathbf{H}^\top \mathbf{H}x - \mathbf{H}^\top \mathbf{y} \quad (29)$$

where  $\mathbf{H}$  is a block-diagonal matrix with element distribution  $\{0, 1\}$ , all elements of observation  $\mathbf{y}$  are non-negative, so the result of  $\mathbf{H}^\top \mathbf{y}$  is non-negative.  $\mathbf{H}^\top \mathbf{H}x$  can be viewed as a weighted sum of  $x$ , so  $\left\| \mathbf{H}^\top \mathbf{H}x \right\|_2 \leq n_x n_y N \left\| x \right\|_2$ , since all elements in  $x$  are normalized to be between 0 and 1, the inequality can be simplified to:  $\left\| \mathbf{H}^\top \mathbf{H}x \right\|_2 \leq n_x n_y N$ . Therefore, the assumption of constant  $M$ ,  $\left\| \nabla f(x) \right\|_2 \leq M$ , bounded gradient holds in the problem of CUP reconstruction. According to the proof of [24], the CUP reconstruction algorithm based on PnP-ADMM has fixed-point convergence, that is, there is  $(x^*, v^*, u^*)$ , and when  $k \rightarrow \infty$ , we have:

$$\left\| x^{(k)} - x^* \right\|_2 \rightarrow 0 \quad (30)$$

$$\left\| v^{(k)} - v^* \right\|_2 \rightarrow 0 \quad (31)$$

$$\left\| u^{(k)} - u^* \right\|_2 \rightarrow 0 \quad (32)$$

## 4. Experiment Results

### 4.1. PSNR and SSIM on Simulation Datasets

In order to test the reconstruction ability of PnP-FFDNet, eight frames of data were selected from the benchmark datasets *runner*, *kobe*, *traffic*, *drop*, and *crash* [25] to be compressed and encoded in the way of CUP data compression, and then PnP-ADMM was used for reconstruction. A comparison is made with the CUP reconstruction algorithm TwIST used in [4] and other denoising algorithms used in the algorithmic framework of PnP-ADMM: TVD/BM3D/IRCnn/DnCNN. The computing platform configuration parameters we used are as follows: CPU is 12th Gen Intel(R) Core(TM) i7-12700H 2.70 GHz, GPU is NVIDIA GeForce RTX 3060 laptop GPU.

The benchmark datasets used in the experiment are all  $256 \times 256 \times 8$ , the size of each frame is  $256 \times 256$ , and the total number of frames is eight frames. In all subsequent experiments, the offset  $s_0$  of each frame is 1. The dimension of the simulated observation data obtained through the data compression model of CUP is  $263 \times 256$ . The encoding mask size is  $256 \times 256$ , its elements  $\{0, 1\}$  are randomly distributed, and the sampling rate is 50%. The regular term of the TwIST algorithm selects the total variation (TV) norm of the image, and the denoising algorithm selects the TVD. Based on the actual test experience, when the parameters manually adjusted in the algorithm are set as follows, a better CUP reconstruction effect can be achieved. For the TwIST algorithm, the regularization parameter is set to 0.05 and the loop is excited when the error of the objective function of two adjacent loops is less than  $1e-5$ . The regularization parameter  $\rho$  in the PnP-ADMM algorithm is set to 1, and the iteration is excited when  $\Delta_{k+1}$  is less than or equal to  $1e-3$ , where  $\Delta_{k+1}$  is [24]:

$$\Delta_{k+1} = \frac{1}{\sqrt{n}} \left( \|x^{(k+1)} - x^{(k)}\|_2 + \|v^{(k+1)} - v^{(k)}\|_2 + \|u^{(k+1)} - u^{(k)}\|_2 \right) \quad (33)$$

The experiment uses PSNR and SSIM to evaluate the reconstruction performance. PSNR is based on the error between the reconstructed image and the corresponding pixel of the original image. SSIM measures the structural similarity between the reconstructed image and the original image from three aspects: brightness, contrast, and structure. Tables 1–3 summarize the PSNR, SSIM, and execution time, respectively of the reconstruction results of the six algorithms. As learning-based denoising algorithms are implemented based on the open-source framework Pytorch, these denoisers can use GPUs to accelerate forward inference. “use GPU” in Table 3 represents the result of the algorithm using GPU-accelerated computing. Figure 8 shows the reconstruction performance of different algorithms on the benchmark dataset.

**Table 1.** The average PSNR (dB) results.

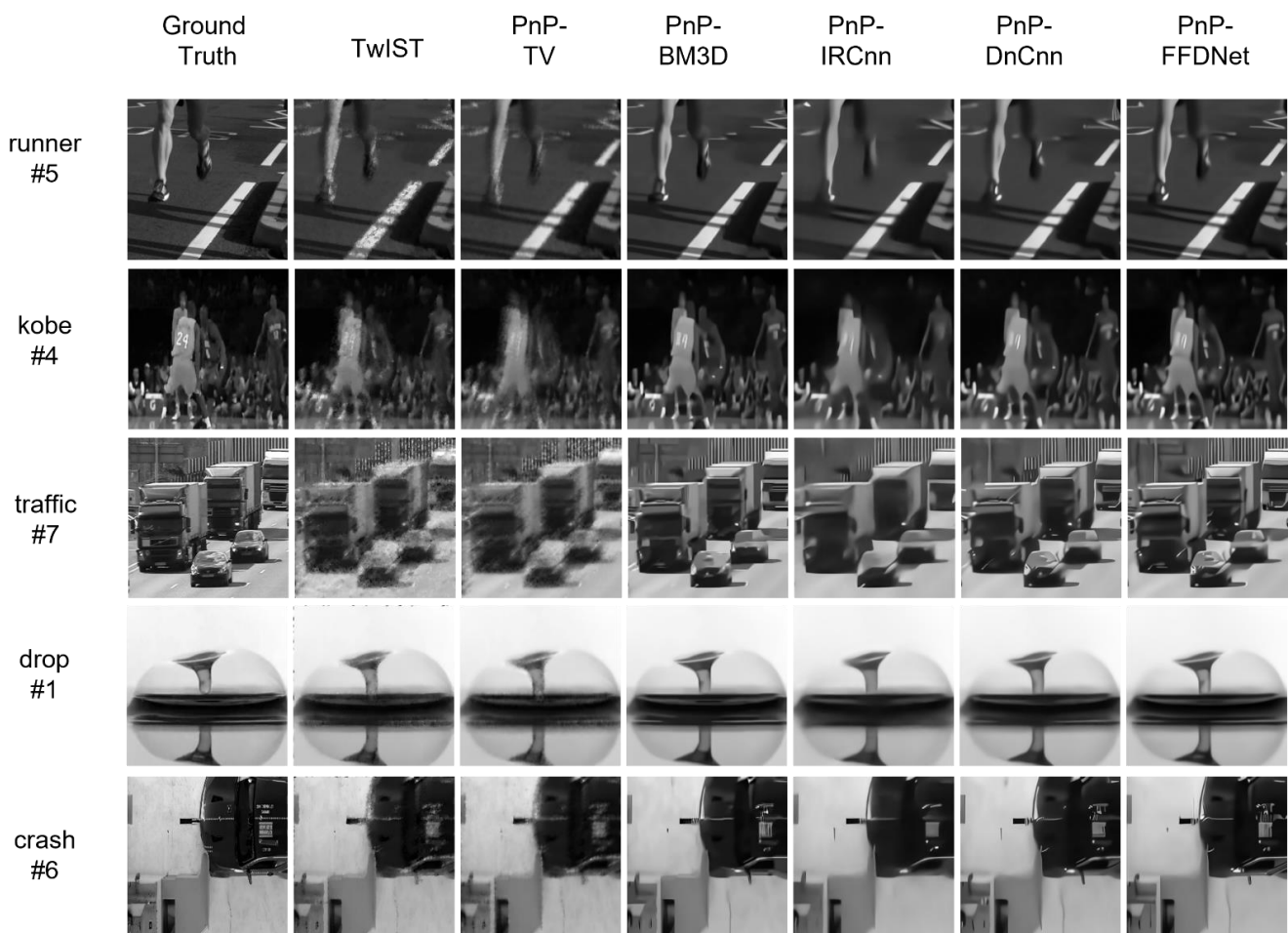
Algorithm	Runner	Kobe	Traffic	Drop	Crash	Average
TwIST	24.52	25.42	19.16	29.39	24.54	24.70
PnP-TV	23.29	23.89	19.55	29.72	24.57	24.20
PnP-BM3D	30.59	29.15	23.77	36.40	26.01	29.18
PnP-IRCnn	25.86	24.83	21.13	29.90	24.56	25.24
PnP-DnCNN	28.12	27.54	22.27	32.60	24.94	27.09
PnP-FFDNet	29.68	28.86	23.19	34.89	25.21	28.37

**Table 2.** The average SSIM results.

Algorithm	Runner	Kobe	Traffic	Drop	Crash	Average
TwIST	0.82	0.82	0.58	0.92	0.82	0.79
PnP-TV	0.82	0.84	0.68	0.95	0.87	0.83
PnP-BM3D	0.95	0.92	0.84	0.98	0.90	0.92
PnP-IRCnn	0.87	0.82	0.73	0.94	0.85	0.84
PnP-DnCNN	0.91	0.85	0.80	0.96	0.87	0.88
PnP-FFDNet	0.93	0.92	0.82	0.98	0.88	0.91

**Table 3.** The execution time (second).

Algorithm	Runner	Kobe	Traffic	Drop	Crash	Average
TwIST	41	67	47	177	104	87
PnP-TV	10	7	9	6	8	8
PnP-BM3D	378	402	387	436	379	396
PnP-IRCnn	33	35	36	34	35	35
PnP-IRCnn (use GPU)	10	11	12	10	11	11
PnP-DnCnn	75	79	80	78	82	79
PnP-DnCnn (use GPU)	14	14	15	14	14	14
PnP-FFDNet	27	27	27	26	27	27
PnP-FFDNet (use GPU)	12	13	12	12	10	12

**Figure 8.** Reconstruction performance of different reconstruction algorithms.

From the statistical results in Tables 1–3, it can be intuitively seen that PnP-FFDNet and PnP-BM3D have the best reconstruction performance. PnP-FFDNet greatly reduces the time required for reconstruction without losing the reconstruction performance. Although other algorithms (such as PnP-TV/PnP-IRCnn/PnP-DnCnn) have advantages in reconstruction time, they sacrifice reconstruction efficiency. Compared with PnP-BM3D, the execution time of PnP-FFDNet is reduced by an average of 93% on the CPU and 96% on the GPU, but the PSNR and SSIM metrics are very similar.

Figure 8 shows the reconstruction performance with different reconstruction algorithms, and selected one frame from eight frames of data for comparison. Among them,

the drop data has fewer details than the other four data images, so the six algorithms have good reconstruction performance. For traffic data with more image texture, although the performance of the six reconstruction algorithms is relatively poor, the reconstruction results of the PnP-FFDNet and PnP-BM3D algorithms have clearer contours and less noise.

#### 4.2. The Performance of PnP-FFDNet on Data with Different Compression Ratios

In order to test the reconstruction performance of the algorithm with different compression ratios, different frame numbers were intercepted from the drop dataset for CUP data compression encoding, and the PnP-FFDNet algorithm and the PnP-BM3D algorithm were selected for comparison. The data compression ratio  $R$  is defined as:

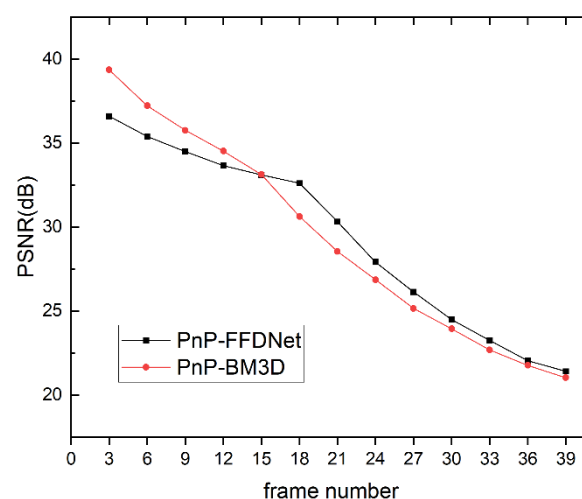
$$R = \frac{\text{size}(x)}{\text{size}(y)} = \frac{n_x n_y N}{Ln_y} = \frac{n_x N}{L} = \frac{n_x N}{n_x + s_0(N - 1)} \quad (34)$$

Since the size of each frame in the *drop* dataset is  $256 \times 256$  and  $s_0$  is 1, the CUP data compression ratio  $R$  can be simplified as:

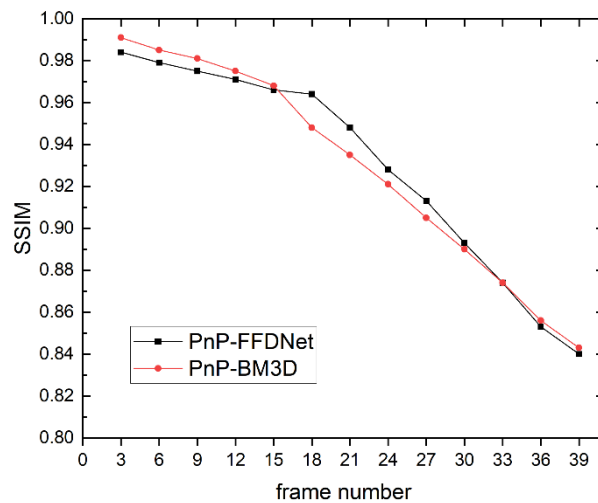
$$R = \frac{256N}{N + 255} \quad (35)$$

The larger the number of frames selected for compression coding, the larger the data compression ratio.

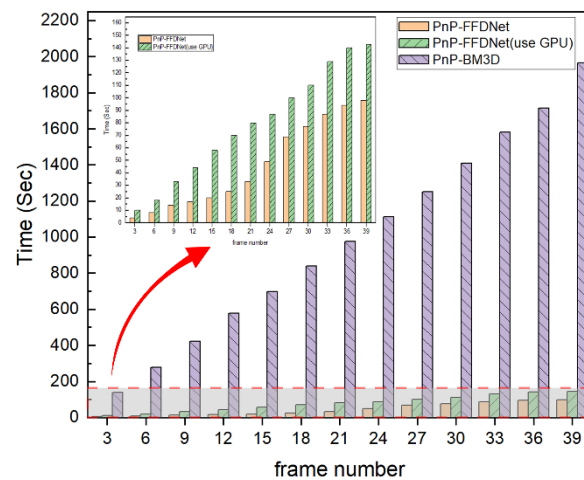
As shown in Figures 9–11, when the data compression rate  $R$  increases, our experimental results show that the reconstructed metrics PSNR and SSIM both decrease. When the compression ratio is small, the reconstruction effect of PnP-BM3D is slightly better than that of PnP-FFDNet, but as the compression ratio  $R$  increases, the performance of the two algorithms in PSNR and SSIM is close, and even when the compression ratio  $R$  is at a certain value, PnP-FFDNet performs better than PnP-BM3D on PSNR and SSIM. However, when the compression ratio  $R$  increases, the time consumption of the PnP-BM3D algorithm increases linearly. When the number of compressed frames is 39, the algorithm takes half an hour to complete the reconstruction, while PnP-FFDNet completes the reconstruction in one minute, due to the high efficiency of the algorithm itself and the parallel acceleration of the GPU. The inference speed of FFDNet is greatly accelerated due to the use of down-sampling techniques to reduce the computational load. Therefore, even in CPU, PnP-FFDNet is significantly faster than PnP-BM3D. Furthermore, accelerating the inference phase of the network with the help of GPU parallel computing is also one of the reasons why the PnP-FFDNet algorithm executes faster.



**Figure 9.** Line chart of PSNR of the reconstruction results of PnP-FFDNet and PnP-BM3D when the number of CUP compressed frames increases.



**Figure 10.** Line chart of SSIM of the reconstruction results of PnP-FFDNet and PnP-BM3D when the number of CUP compressed frames increases.

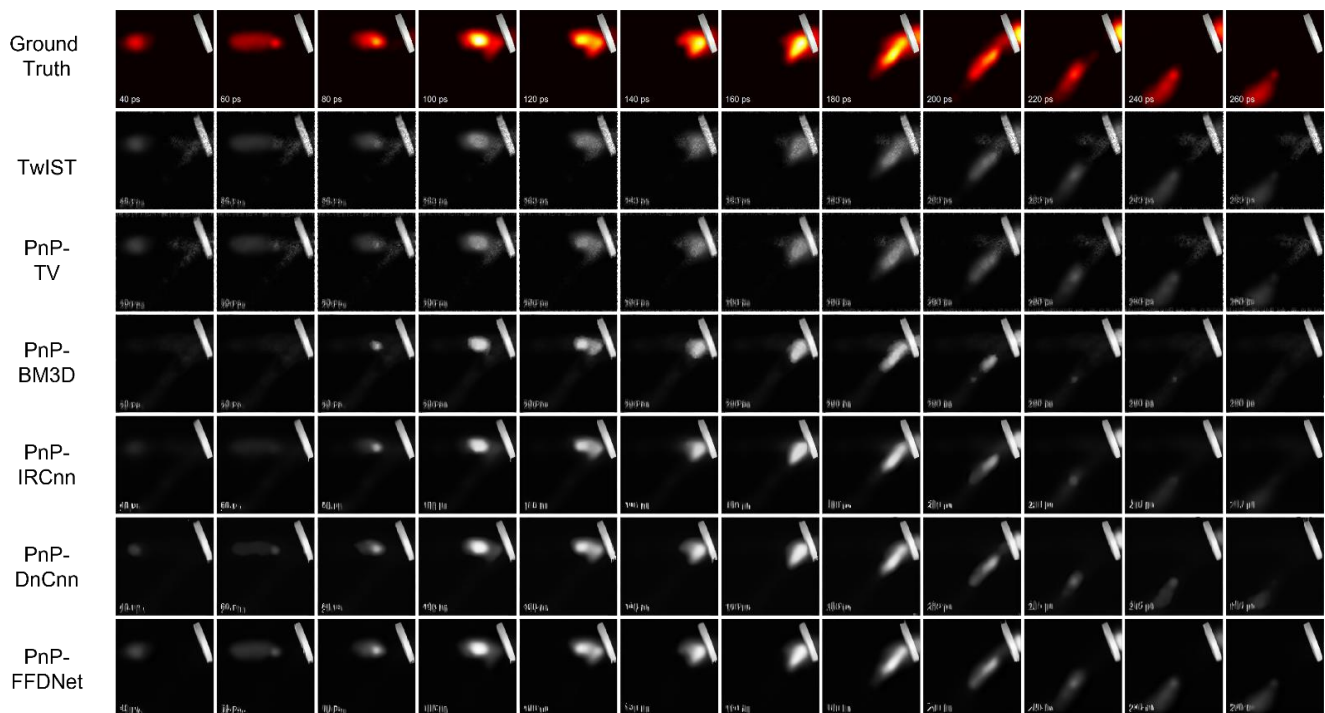


**Figure 11.** Histogram of execution time of PnP-BM3D, PnP-FFDNet, and PnP-FFDNet (use GPU) when the number of compressed frame increases. The subplot in Figure 7 is an enlarged histogram of the red dashed area.

#### 4.3. Performance of PnP-FFDNet on Real Data

In order to test the performance of the PnP-FFDNet algorithm on real CUP experimental data, key frames were extracted from the video file of the laser pulse reflection process in the Supplementary Material of [4], and the RGB image was converted into a grayscale image to form a  $320 \times 320 \times 16$  original image data, and  $335 \times 320$  observation data were obtained through the CUP data compression model. Using TwIST and PnP-TV/BM3D/IRCnn/DnCNN/FFDNet, we performed algorithm reconstruction experiments.

Figure 12 shows the reconstruction results with six algorithms on the real experimental data of CUP. The algorithm reconstructs a total of 16 frames, and selects 12 consecutive frames with clear pulsed laser graphics for comparison. Due to the process of compression sampling, only part of the information is sampled, the spatial resolution of the image reconstructed by the algorithm is low, and so the text in the image has more details and, therefore, cannot be reconstructed well. From the image comparison of the reconstruction results, it can be seen that the laser reflection process of the PnP-FFDNet reconstruction results is the closest to the ground truth, and the reconstructed visual effects, evaluation indicators (PSNR, SSIM), and operating efficiency all exceed the other algorithms.



**Figure 12.** Reconstruction performance of different reconstruction algorithms on real data.

Table 4 shows the reconstruction indicators. Compared with TwIST, PnP-TV and PnP-BM3D, PSNR is improved by 3.95 dB, 4.61 dB, and 1.85 dB, respectively, and SSIM is improved by 0.07, 0.1, and 0.02, respectively. Under the premise of obtaining a better reconstruction effect, PnP-IRCnn/PnP-DnCNN/PnP-FFDNet require relatively less reconstruction time. The reconstruction effects of the algorithms using the learning-based denoiser are similar on the metric PSNR, and PnP-FFDNet is the best for the metric SSIM. Although the running time of the PnP-TV algorithm is the shortest, its reconstruction effect is similar to that of TwIST. The performance on real datasets shows that PnP-FFDNet has excellent performance in reconstruction effect and running efficiency.

**Table 4.** Results of PSNR (dB), SSIM, execution time (s) and execution time (s) (use GPU) for different algorithms on real data.

Algorithm	PSNR	SSIM	Execution Time (s)	Execution Time (s) (Use GPU)
TwIST	21.17	0.88	270	-
PnP-TV	21.04	0.85	33	-
PnP-BM3D	23.80	0.93	1124	-
PnP-IRCnn	25.67	0.91	103	26
PnP-DnCNN	25.27	0.87	230	40
PnP-FFDNet	25.65	0.95	85	43

## 5. Conclusions

This paper proposes a CUP reconstruction algorithm based on the combination of the PnP-ADMM framework and the convolutional neural network denoising algorithm FFDNet. The reconstruction methods were performed on the benchmark dataset and the real data of the CUP experiment. The results show that the proposed algorithm performs better on the PSNR and SSIM. Since the inference of the convolutional neural network can be performed in parallel using the GPU, the execution time of PnP-FFDNet is greatly reduced compared to PnP-BM3D, without losing the re-estimation effect. In the popular CUP experimental configuration, the data depth is often between 150–1500. The proposed

method can greatly speed up the reconstruction process. and is a practical and efficient CUP reconstruction algorithm.

**Author Contributions:** Formal analysis, Q.S.; methodology, Q.S.; supervision, J.T.; writing—original draft, Q.S.; writing—review and editing, J.T. and C.P. All authors have read and agreed to the published version of the manuscript.

**Funding:** The work was supported by the Scientific Instrument Developing Project of the Chinese Academy of Sciences, Grant No. GJJSTD20220006.

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Informed consent was obtained from all subjects involved in the study.

**Data Availability Statement:** Not applicable.

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

1. Eldar, Y.C.; Kutyniok, G. *Compressed Sensing: Theory and Applications*; Cambridge University Press: Cambridge, UK, 2012.
2. Liang, J.; Wang, L.V. Single-shot ultrafast optical imaging. *Optica* **2018**, *5*, 1113–1127. [[CrossRef](#)] [[PubMed](#)]
3. Tsia, K.K.; Goda, K.; Thompson, J.V.; Mason, J.D.; Beier, H.T.; Bixler, J.N. High speed fluorescence imaging with compressed ultrafast photography. In Proceedings of the Society of Photo-Optical Instrumentation Engineers (SPIE), San Francisco, CA, USA, 28 January–2 February 2017; p. 1007613.
4. Gao, L.; Liang, J.; Li, C.; Wang, L.V. Single-shot compressed ultrafast photography at one hundred billion frames per second. *Nature* **2014**, *516*, 74–77. [[CrossRef](#)] [[PubMed](#)]
5. Liang, J.; Ma, C.; Zhu, L.; Chen, Y.; Gao, L.; Wang, L.V. Single-shot real-time video recording of a photonic Mach cone induced by a scattered light pulse. *Sci. Adv.* **2017**, *3*, e1601814. [[CrossRef](#)] [[PubMed](#)]
6. Liang, J.; Gao, L.; Hai, P.; Li, C.; Wang, L. Encrypted three-dimensional dynamic imaging using snapshot time-of-flight compressed ultrafast photography. *Sci. Rep.* **2015**, *5*, 15504. [[CrossRef](#)] [[PubMed](#)]
7. Bioucas-Dias, J.M.; Figueiredo, M.A.T. A new TwIST: Two-step iterative shrinkage/thresholding algorithms for image restoration. *IEEE Trans. Image Process.* **2007**, *16*, 2992–3004. [[CrossRef](#)] [[PubMed](#)]
8. Rudin, L.I.; Osher, S.; Fatemi, E. Nonlinear total variation based noise removal algorithms. *Phys. D Nonlinear Phenom.* **1992**, *60*, 259–268. [[CrossRef](#)]
9. Zhu, L.; Chen, Y.; Liang, J.; Xu, Q.; Gao, L.; Ma, C.; Wang, L.V. Space- and intensity-constrained reconstruction for compressed ultrafast photography. *Optica* **2016**, *3*, 694–697. [[CrossRef](#)] [[PubMed](#)]
10. Lai, Y.; Xue, Y.; Cté, C.; Liu, X.; Laramée, A.; Jaouen, N.; Légaré, F.; Tian, L.; Liang, J. Compressed ultrafast photography: Single-shot ultraviolet compressed ultrafast photography (laser photonics rev. 14(10)/2020). *Laser Photonics Rev.* **2020**, *14*, 2070055. [[CrossRef](#)]
11. Venkatakrishnan, S.V.; Bouman, C.A.; Wohlberg, B. Plug-and-play priors for model based reconstruction. In Proceedings of the 2013 IEEE Global Conference on Signal and Information Processing, Austin, TX, USA, 3–5 December 2013.
12. Dabov, K.; Foi, A.; Katkovnik, V.; Egiazarian, K. Image denoising by sparse 3-D transform-domain collaborative filtering. *IEEE Trans. Image Process.* **2007**, *16*, 2080–2095. [[CrossRef](#)] [[PubMed](#)]
13. Jia, R.; Zhao, H. A fast algorithm for the total variation model of image denoising. *Adv. Comput. Math.* **2010**, *33*, 231–241. [[CrossRef](#)]
14. Gu, S.; Lei, Z.; Zuo, W.; Feng, X. Weighted nuclear norm minimization with application to image denoising. In Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Columbus, OH, USA, 23–28 June 2014.
15. Mairal, J.; Elad, M.; Sapiro, G. Sparse representation for color image restoration. *IEEE Trans. Image Process.* **2007**, *17*, 53–69. [[CrossRef](#)] [[PubMed](#)]
16. Elad, M.; Aharon, M. Image denoising via sparse and redundant representations over learned dictionaries. *IEEE Trans. Image Process.* **2006**, *15*, 3736–3745. [[CrossRef](#)] [[PubMed](#)]
17. Dong, W.; Lei, Z.; Shi, G. Nonlocally centralized sparse representation for image restoration. *IEEE Trans. Image Process.* **2013**, *22*, 1620–1630. [[CrossRef](#)] [[PubMed](#)]
18. Mairal, J.; Bach, F.; Ponce, J.; Sapiro, G.; Zisserman, A. Non-local sparse models for image restoration. In Proceedings of the 2009 IEEE 12th International Conference on Computer Vision (ICCV), Kyoto, Japan, 29 September–2 October 2010.
19. Buades, A.; Coll, B.; Morel, J.M. A non-local algorithm for image denoising. In Proceedings of the 2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05), San Diego, CA, USA, 20–25 June 2005.
20. Kai, Z.; Zuo, W.; Gu, S.; Lei, Z. Learning deep CNN denoiser prior for image restoration. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017.
21. Zhang, K.; Zuo, W.; Chen, Y.; Meng, D.; Zhang, L. Beyond a gaussian denoiser: Residual learning of deep CNN for image denoising. *IEEE Trans. Image Process.* **2017**, *26*, 3142–3155. [[CrossRef](#)] [[PubMed](#)]

22. Zhang, K.; Zuo, W.; Zhang, L. FFDNet: Toward a fast and flexible solution for CNN-based image denoising. *IEEE Trans. Image Process.* **2018**, *27*, 4608–4622. [[CrossRef](#)] [[PubMed](#)]
23. Tassano, M.; Delon, J.; Veit, T. An Analysis and Implementation of the FFDNet Image Denoising Method. *Image Processing On Line* **2019**, *9*, 1–25. [[CrossRef](#)]
24. Chan, S.H.; Wang, X.; Elgandy, O.A. Plug-and-play ADMM for image restoration: Fixed-point convergence and applications. *IEEE Trans. Comput. Imaging* **2016**, *3*, 84–98. [[CrossRef](#)]
25. Yuan, X.; Liu, Y.; Suo, J.; Dai, Q. Plug-and-play algorithms for large-scale snapshot compressive imaging. In Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 14–19 June 2020.
26. Sreehari, S.; Venkatakrisnan, S.V.; Wohlberg, B.; Buzzard, G.T.; Drummy, L.F.; Simmons, J.P.; Bouman, C.A. Plug-and-play priors for bright field electron tomography and sparse interpolation. *IEEE Trans. Comput. Imaging* **2016**, *2*, 408–423. [[CrossRef](#)]
27. Rond, A.; Giryes, R.; Elad, M. Poisson inverse problems by the plug-and-play scheme. *J. Vis. Commun. Image Represent.* **2016**, *41*, 96–108. [[CrossRef](#)]
28. Dar, Y.; Bruckstein, A.M.; Elad, M.; Giryes, R. Postprocessing of compressed images via sequential denoising. *IEEE Trans. Image Process.* **2016**, *25*, 3044–3058. [[CrossRef](#)] [[PubMed](#)]
29. Brifman, A.; Romano, Y.; Elad, M. Turning a denoiser into a super-resolver using plug and play priors. In Proceedings of the 2016 IEEE International Conference on Image Processing (ICIP), Phoenix, AZ, USA, 25–28 September 2016; pp. 1404–1408. [[CrossRef](#)]