ACCELERATION OF PDS-BASED HIGH-DIMENSIONAL SIGNAL RESTORATION

Gai YAMAMOTO*, Yuya KODAMA*, Shogo MURAMATSU†, Samuel CHOI† and Gwanggil JEON‡
* Graduate School of Science and Technology, Niigata Univ., Japan
† Faculty of Engineering, Niigata Univ., Japan
‡ Dept. of Embedded Systems Engineering, Incheon National Univ., Korea

***Abstract*—In this study, acceleration techniques for the primal-dual splitting (PDS) algorithm, one of the optimization algorithms used to restore high-dimensional signals, are proposed. In general, it is inevitable that signals observed by sensors are incompletely measured and/or contaminated by noise. Thus, there is a demand for techniques to restore signals obtained in inadequate environments. Restoration of high-dimensional signals such as volumetric data often takes several hours or even days. Therefore, speeding up the restoration process is an important issue. This study addresses this issue in two different ways. First, acceleration methods are introduced into PDS. Second, their fixed-point implementations are introduced. In order to verify the significance of the proposed approach, the restoration performance and processing speed are evaluated and compared with previous techniques.**

I. INTRODUCTION

Today, with the advancement of imaiging technology, image data have developed from two-dimensional (2-D) to three-dimensional (3-D). As well, the resolution of images has increased, and the volume have become larger. Particularly in medical diagnosis and bio-science, magnetic resonance imaging (MRI), computed tomography (CT), and optical coherence tomography (OCT) are used to observe the brain, internal organs, and other parts of living bodies without actually making incisions, contributing to the early detection of diseases and the elucidation of biological functions [1]. On the other hand, it is unavoidable that signals captured by sensors are generally affected by noise and degradation caused by measurement devices to some extent. For this type of sensing, signal processing is essential to remove noise caused during the observation process and to recover signals measured under tough environments.

Convex optimization is one of the methods used in the restoration process. Among convex optimiation algorithms, the proximal splitting approachs are known to be able to efficiently handle relatively large problems with proximity operators. For this reason, it is widely used in the fields of image restoration and compressed sensing. Typical proximal splitting optimization algorithms include the primal-dual splitting method (PDS) [2]–[6].

The demand for processing of large amount signals such as volumetric data and high-resolution images is increasing. With the increase in signal volume, large amounts of data may have to be processed by computers. For example, in the article [7], OCT data of size $256 \times 256 \times 1673$ voxels are processed.

Algorithm 1 Primal-Dual Splitting (PDS) Algorithm

Input: $\boldsymbol{x}^{(0)}, \boldsymbol{y}^{(0)}, \gamma_1 > 0, \gamma_2 > 0, n \leftarrow 0$
Output: $\boldsymbol{x}^{(n)}$

1: **while** A stopping criterion is not satisfied **do**
2: $\quad \boldsymbol{x}^{(n+1)} = \mathrm{prox}_{\gamma_1 g}(\boldsymbol{x}^{(n)} - \gamma_1(\nabla f(\boldsymbol{x}^{(n)}) + \boldsymbol{G}^\top \boldsymbol{y}^{(n)})$
3: $\quad \boldsymbol{y}^{(n+1)} = \mathrm{prox}_{\gamma_2 h^*}(\boldsymbol{y}^{(n)} + \gamma_2 \boldsymbol{G}(2\boldsymbol{x}^{(n+1)}) - \boldsymbol{x}^{(n)})$
4: $\quad n \leftarrow n + 1$
5: **end while**

As the amount of computation increases, so does the time required for processing, which can take several hours or even days. In the restoration process of high-resolution and high-dimensional data, it is essential to use an acceleration method that is appropriate for the observed signal, thereby reducing processing time and computational resources.

Therefore, in this work, we propose to accelerate PDS in order to solve the processing time problem. We address this issue in two different ways. First, we propose to introduce an acceleration method into the gradient descent step of PDS in order to reduce the computation time. As acceleration methods for PDS, we adopt RMSpropGraves [8] and Adam [9]. By simulating the OCT image restoration, we confirm the reduction of computation time and evaluate the restoration performance. Second, we propose to reduce the computation time by introducing fixed-point arithmetic. By simulating the restoration of two-dimensional color images, we confirm the reduction of computation time and evaluate the restoration performance.

II. REVIEW OF IMAGE RESTORATION

One existing signal restoration method is the convex optimization method based on proximal splitting. By setting the image restoration problem as a convex optimization problem, it can be solved by the iterative algorithm. PDS is one of the typical proximity splitting optimization algorithms.

A. Primal-Dual Splitting (PDS) Method

Let us consider an optimization problem,

$$\underset{\boldsymbol{x}\in\mathbb{R}^N}{\mathrm{argmin}} f(\boldsymbol{x}) + g(\boldsymbol{x}) + h(\boldsymbol{G}\boldsymbol{x}), \tag{1}$$

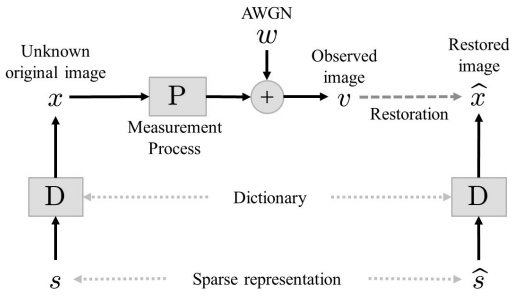


Fig. 1. Restoration model based on sparse representation corresponding to the degradation model as in (3)

where $f, g : \mathbb{R}^N \to \mathbb{R} \cup \{\infty\}$ and $h : \mathbb{R}^M \to \mathbb{R} \cup \{\infty\}$, and f is a function that is differentiable and its gradient ∇f is Lipschitz continuous, g and h are closed convex functions that are lower semicontinuous and proper, and $\boldsymbol{G} \in \mathbb{R}^{M \times N}$. PDS is an algorithm for solving the optimization problem in (1). One realization of PDS algorithm is shown in Algorithm 1 [2]–[6], where $\gamma_1, \gamma_2 > 0$ are step size parameters satisfying

$$\gamma_1 \left(\frac{\beta}{2} + \gamma_2 (\sigma_{\max}(\boldsymbol{G}^\top \boldsymbol{G})) \right) < 1, \tag{2}$$

where $\sigma_{\max}(\cdot)$ denotes the maximum singular value. PDS avoids inverse operations in the computation process and can solve problems with a large number of functions and constraints. These features make PDS suitable for high-dimensional signal recovery.

B. Signal Restoration Model

Let us consider the degradation model

$$\boldsymbol{v} = \boldsymbol{P}\boldsymbol{x} + \boldsymbol{w}, \tag{3}$$

where $\boldsymbol{v} \in \mathbb{R}^M$ is an observed signal, $\boldsymbol{x} \in \mathbb{R}^N$ is an unknown original signal, $\boldsymbol{P} \in \mathbb{R}^{M \times N}$ is a measurement process, and $\boldsymbol{w} \in \mathbb{R}^M$ is an additive white Gaussian noise (AWGN).

In the restoration process of high-dimensional signals such as image and volumetric data, it is important to have a generative model that can efficiently represent the original data in order to achieve high quality signal restoration. An example of a generative model is the use of a dictionary, representing the original data $\boldsymbol{x}$ as $\boldsymbol{x} = \boldsymbol{D}\boldsymbol{s}$, a product of matrix $\boldsymbol{D} \in \mathbb{R}^{N \times L}$ and coefficient vector $\boldsymbol{s} \in \mathbb{R}^L$. In this model, we can set matrix $\boldsymbol{D}$ such that $\boldsymbol{s}$ is almost zero (sparse). The set of column vectors of $\boldsymbol{D}$ is called a dictionary. Dictionaries include DCT, DWT and undecimated Haar transform (UDHT) [10]–[12]. By selecting an appropriate dictionary, we can achieve high quality restoration. The restoration model based on the sparse representation with dictionary $\boldsymbol{D}$ is represented as in Fig. 1. The restored signal can be estimated as $\hat{\boldsymbol{x}} = \boldsymbol{D}\hat{\boldsymbol{s}}$ [13], [14].

III. Proposed Acceleration

In this section, we propose two approaches to accelerate the signal restoration based on PDS. The first approach is the acceleration of the gradient descent step in PDS and the second one is the implementation with the fixed-point arithmetic.

Algorithm 2 PDS with RMSpropGraves

Input: $\boldsymbol{x}^{(0)}, \boldsymbol{y}^{(0)}, \boldsymbol{m}^{(0)}, \boldsymbol{v}^{(0)}$
$\gamma_1, \gamma_2, \rho_1, \epsilon > 0, n \leftarrow 0$
Output: $\boldsymbol{x}^{(n)}$

while A stopping criterion is not satisfied **do**
  $\boldsymbol{g}^{(n)} = \nabla f(\boldsymbol{x}^{(n)})$
  $\boldsymbol{m}^{(n+1)} = \rho_1 \boldsymbol{m}^{(n)} + (1-\rho_1)\boldsymbol{g}^{(n)}$
  $\boldsymbol{v}^{(n+1)} = \rho_1 \boldsymbol{v}^{(n)} + (1-\rho_1)\boldsymbol{g}^{(n)} \odot \boldsymbol{g}^{(n)}$
  $\boldsymbol{w}^{(n)} = (\boldsymbol{v}^{(n+1)} - (\boldsymbol{m}^{(n+1)})^{\circ 2} + \epsilon)^{\circ -\frac{1}{2}} \odot \boldsymbol{g}^{(n)}$
  $\boldsymbol{x}^{(n+1)} = \mathrm{prox}_{\gamma_1 g}(\boldsymbol{x}^{(n)} - \gamma_1(\boldsymbol{w}^{(n)} + \boldsymbol{G}^\top \boldsymbol{y}^{(n)}))$
  $\boldsymbol{y}^{(n+1)} = \mathrm{prox}_{\gamma_2 h^*}(\boldsymbol{y}^{(n)} + \gamma_2 \boldsymbol{G}(2\boldsymbol{x}^{(n+1)}) - \boldsymbol{x}^{(n)}))$
  $n \leftarrow n+1$
end while

A. Gradient descent acceleration

Let us propose to introduce an acceleration method into PDS [15]. Note that the second step

$$\boldsymbol{x}^{(n+1)} = \mathrm{prox}_{\gamma_1 g}\left(\boldsymbol{x}^{(n)} - \gamma_1 \nabla f(\boldsymbol{x}^{(n)}) - \gamma_1 \boldsymbol{G}^\top \boldsymbol{y}^{(n)} \right) \tag{4}$$

of the PDS algorithm shown in Algorithm 1 contains the gradient descent, $\boldsymbol{x}^{(n)} - \gamma_1 \nabla f(\boldsymbol{x}^{(n)})$. Therefore, we can accelerate the convergence to the optimal solution by adaptively adjusting the learning rate γ_1. In this paper, we introduce RMSpropGraves, which is a modified algorithm of RMSprop (see Appendix A), and Adam, which is one of the most widely used acceleration methods combining RMSProp and momentum methods (see Appendix B).

Algorithm 2 shows a variant of Algorithm 1 with RMSpropGraves, where ρ_1 is the decay rate indicating how much of the gradient information in the $(n-1)$-th iteration is used, the superscript "$\circ$" is the Hadamard power, $\odot$ is the Hadamard product, and both of $\boldsymbol{v}^{(0)}$ and $\boldsymbol{m}^{(0)}$ are the zero matrices. ϵ is a value to avoid zero division, and should be set to an extremely small value such as $\epsilon = 1.0 \times 10^{-6}$. ρ_1 is an exponential decay parameter for moment estimation, often set to around $\rho_1 = 0.95$ in RMSPropGraves.

B. Fixed-point arithmetic

In this study, we also propose to accelerate PDS by fixed-point arithmetic. In general, there are two types of arithmetic operations in computers: floating-point arithmetic and fixed-point arithmetic. While the floating-point number type can represent a wide range of values from large to small, it requires higher computation cost than the fixed-point number type. Fixed-point arithmetic is introduced in consideration of its impact on the quality of restoration as a means of speeding up computation.

Algorithm 3 PDS for (6)

Input: $\boldsymbol{x}^{(0)}, \boldsymbol{y}_1^{(0)}, \boldsymbol{y}_2^{(0)}, \gamma_1 > 0, \gamma_2 > 0, n \leftarrow 0$
Output: $\boldsymbol{x}^{(n)}, \boldsymbol{q}^{(n)}$
$\boldsymbol{q}^{(0)} = \boldsymbol{D}\boldsymbol{x}^{(0)}$
while A stopping criterion is not satisfied **do**
  $\boldsymbol{t} \leftarrow \boldsymbol{D}^\intercal(\nabla f(\boldsymbol{q}^{(n)}) + \boldsymbol{\Delta_z}^\intercal \boldsymbol{y_1}^{(n)} + \boldsymbol{y_2}^{(n)})$
  $\boldsymbol{x}^{(n+1)} = \mathfrak{G}_{\lambda\|\cdot\|_1}(\boldsymbol{x}^{(n)} - \gamma_1 \boldsymbol{t}, \gamma_1^{\frac{1}{2}})$
  $\boldsymbol{q}^{(n+1)} = \boldsymbol{D}\boldsymbol{x}^{(n+1)}$
  $\boldsymbol{u} \leftarrow 2\boldsymbol{q}^{(n+1)} - \boldsymbol{q}^{(n)}$
  $\boldsymbol{y}_1^{(n)} \leftarrow \boldsymbol{y}_1^{(n)} + \gamma_2 \boldsymbol{\Delta_z} \boldsymbol{u}$
  $\boldsymbol{y}_2^{(n)} \leftarrow \boldsymbol{y}_2^{(n)} + \gamma_2 \boldsymbol{u}$
  $\boldsymbol{y}_1^{(n+1)} = \boldsymbol{y}_1^{(n)} - \gamma_2 \mathfrak{G}_{\eta\|\cdot\|_1}(\gamma_2^{-1}\boldsymbol{y}_1^{(n)}, \gamma_2^{-\frac{1}{2}})$
  $\boldsymbol{y}_2^{(n+1)} = \boldsymbol{y}_2^{(n)} - \gamma_2 \mathcal{P}_{[a,b]^N}(\gamma_2^{-1}\boldsymbol{y}_2^{(n)})$
  $n \leftarrow n + 1$
end while

IV. Performance evaluation

In this section, we first evaluate the performance by the 3-D OCT restoration process to confirm the effectiveness of the implementation of the acceleration method of the gradient descent step [7]. Next, to confirm the effectiveness of the implementation of fixed-point arithmetic, we evaluate the restoration performance for the Kodak Lossless True Color Image set by using total variation (TV) regularization [16].

A. Evaluation of gradient descent acceleration

In this section, we apply the proposed acceleration method to the 3-D OCT data restoration process proposed in [7], and evaluate the processing time and restoration performance.

1) Problem setting: The OCT observation process is modeled using a coherence function. The coherence function has a shape similar to a cosine-modulated Gaussian function as shown in Fig. 2. Let $\boldsymbol{P}$ be the measurement process through the coherence function.

In the article [7], a synthesis dictionary $\boldsymbol{D} \in \mathbb{R}^{N\times L}$ is assumed to construct a generative model

$$\boldsymbol{u} = \boldsymbol{D}\boldsymbol{s} \tag{5}$$

for the potential refractive index distribution $\boldsymbol{u}$ [17]. The problem setting adopted in [7] is formulated as

$$\hat{\boldsymbol{s}} = \underset{\boldsymbol{s}\in\mathbb{R}^L}{\arg\min} \frac{1}{2}\|\boldsymbol{P}\phi(\boldsymbol{D}\boldsymbol{s}) - \boldsymbol{v}\|_2^2 + \lambda\|\boldsymbol{s}\|_1 + \eta\|\boldsymbol{\Delta_z}\boldsymbol{D}\boldsymbol{s}\|_1, \quad \text{s.t. } \boldsymbol{D}\boldsymbol{s} \in [a,b]^N, \tag{6}$$

where $||\cdot||_1$ is the ℓ_1-norm, $||\cdot||_2$ is the ℓ_2-norm, and $\lambda, \eta \in [0,\infty)$ are the regularization parameters. $\boldsymbol{\Delta_z}$ indicates a difference operation in the Z direction and $\phi : [0,\infty)^N \to (-1,1)^N$ denotes the mapping from refractive index to reflectance, and a and b determine the lower and upper boundary of refractive indices, respectively.

Algorithm 4 PDS with RMSpropGraves for (6)

Input: $\boldsymbol{x}^{(0)}, \boldsymbol{y}_1^{(0)}, \boldsymbol{y}_2^{(0)}, \boldsymbol{m}^{(0)}, \boldsymbol{v}^{(0)}$
  $\gamma_1 > 0, \gamma_2 > 0, \rho_1 > 0, \epsilon > 0, n \leftarrow 0$
Output: $\boldsymbol{x}^{(n)}, \boldsymbol{q}^{(n)}$
$\boldsymbol{q}^{(0)} = \boldsymbol{D}\boldsymbol{x}^{(0)}$
while A stopping criterion is not satisfied **do**
  $\boldsymbol{g}^{(n)} = \nabla f(\boldsymbol{q}^{(n)})$
  $\boldsymbol{m}^{(n+1)} = \rho_1 \boldsymbol{m}^{(n)} + (1-\rho_1)\boldsymbol{g}^{(n)}$
  $\boldsymbol{v}^{(n+1)} = \rho_1 \boldsymbol{v}^{(n)} + (1-\rho_1)(\boldsymbol{g}^{(n)})^{\circ 2}$
  $\boldsymbol{w}^{(n)} = (\boldsymbol{v}^{(n+1)} - (\boldsymbol{m}^{(n+1)})^{\circ 2} + \epsilon)^{\circ -\frac{1}{2}} \odot \boldsymbol{g}^{(n)}$
  $\boldsymbol{t} \leftarrow \boldsymbol{D}^\intercal(\boldsymbol{w}^{(n)} + \boldsymbol{\Delta_z}^\intercal \boldsymbol{y_1}^{(n)} + \boldsymbol{y_2}^{(n)})$
  $\boldsymbol{x}^{(n+1)} = \mathfrak{G}_{\lambda\|\cdot\|_1}(\boldsymbol{x}^{(n)} - \gamma_1 \boldsymbol{t}, \gamma_1^{\frac{1}{2}})$
  $\boldsymbol{q}^{(n+1)} = \boldsymbol{D}\boldsymbol{x}^{(n+1)}$
  $\boldsymbol{u} \leftarrow 2\boldsymbol{q}^{(n+1)} - \boldsymbol{q}^{(n)}$
  $\boldsymbol{y}_1^{(n)} \leftarrow \boldsymbol{y}_1^{(n)} + \gamma_2 \boldsymbol{\Delta_z} \boldsymbol{u}$
  $\boldsymbol{y}_2^{(n)} \leftarrow \boldsymbol{y}_2^{(n)} + \gamma_2 \boldsymbol{u}$
  $\boldsymbol{y}_1^{(n+1)} = \boldsymbol{y}_1^{(n)} - \gamma_2 \mathfrak{G}_{\eta\|\cdot\|_1}(\gamma_2^{-1}\boldsymbol{y}_1^{(n)}, \gamma_2^{-\frac{1}{2}})$
  $\boldsymbol{y}_2^{(n+1)} = \boldsymbol{y}_2^{(n)} - \gamma_2 \mathcal{P}_{[a,b]^N}(\gamma_2^{-1}\boldsymbol{y}_2^{(n)})$
  $n \leftarrow n + 1$
end while

TABLE I
Specifications of OCT data restoration simulation

Simulation Tools	MATLAB R2020b
OS	Ubuntu 16.04.7 LTS
CPU	Intel Xeon E5-2620
Memory	128GB

In order to apply Algorithm 1 to (6), let

$$f(\boldsymbol{x}) = F(\boldsymbol{D}\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{P}\phi_1(\boldsymbol{D}\boldsymbol{x}) - \boldsymbol{v}\|_2^2, \tag{7}$$

$$g(\boldsymbol{x}) = \lambda\|\boldsymbol{x}\|_1, \tag{8}$$

$$h(\boldsymbol{G}\boldsymbol{x}) = \eta\|\boldsymbol{y}_1\|_1 + \iota_{[a,b]^N}(\boldsymbol{y}_2), \tag{9}$$

$$\boldsymbol{G}\boldsymbol{x} = \begin{bmatrix}\boldsymbol{y}_1 \\ \boldsymbol{y}_2\end{bmatrix} = \begin{bmatrix}\boldsymbol{\Delta_z}\boldsymbol{D} \\ \boldsymbol{D}\end{bmatrix}\boldsymbol{x}, \tag{10}$$

where $\phi_1(\cdot)$ is a linear approximation of $\phi(\cdot)$, and $\iota_{[a,b]^N}(\cdot)$ is the indicator function defined by

$$\iota_{[a,b]^N}(\boldsymbol{y}) \triangleq \begin{cases} 0, & \boldsymbol{y} \in [a,b]^N \\ \infty, & \boldsymbol{y} \notin [a,b]^N \end{cases}. \tag{11}$$

Algorithm 3 shows the conventional algorithm for solving (6), where $\mathfrak{G}_{\lambda\|\cdot\|_1}(\boldsymbol{x},\sigma)$ and $\mathcal{P}_{[a,b]^N}(\boldsymbol{x})$ are defined by

$$\left[\mathfrak{G}_{\lambda\|\cdot\|_1}(\boldsymbol{x},\sigma)\right]_n \triangleq \mathrm{sgn}([\boldsymbol{x}]_n)\max\{|[\boldsymbol{x}]_n| - \lambda\sigma^2, 0\}, \tag{12}$$

$$\left[\mathcal{P}_{[a,b]^N}(\boldsymbol{x})\right]_n \triangleq \min\{\max\{[\boldsymbol{x}]_n, a\}, b\}, \tag{13}$$

which are the soft thresholding operation and metric projection, respectively. Both of the operations are element-wise.

The algorithms accelerated by RMSpropGraves [8] and Adam [9] are shown in Algorithms 4 and 5, respectively.

Algorithm 5 PDS with the introduced Adam for (6)

Input: $\boldsymbol{x}^{(0)}, \boldsymbol{y}_1^{(0)}, \boldsymbol{y}_2^{(0)}, \boldsymbol{m}^{(0)}, \boldsymbol{v}^{(0)}$
$\gamma_1 > 0, \gamma_2 > 0, \rho_1 > 0, \rho_2 > 0, \epsilon > 0, n \leftarrow 0$
Output: $\boldsymbol{x}^{(n)}, \boldsymbol{q}^{(n)}$
$\boldsymbol{q}^{(0)} = \boldsymbol{D}\boldsymbol{x}^{(0)}$
while A stopping criterion is not satisfied **do**
  $\boldsymbol{g}^{(n)} = \nabla f(\boldsymbol{q}^{(n)})$
  $\boldsymbol{m}^{(n+1)} = \rho_1 \boldsymbol{m}^{(n)} + (1-\rho_1)\boldsymbol{g}^{(n)}$
  $\boldsymbol{v}^{(n+1)} = \rho_2 \boldsymbol{v}^{(n)} + (1-\rho_2)(\boldsymbol{g}^{(n)})^{\circ 2}$
  $\hat{\boldsymbol{m}}^{(n+1)} = \boldsymbol{m}^{(n)}/(1-\rho_1^{(n+1)})$
  $\hat{\boldsymbol{v}}^{(n+1)} = \boldsymbol{v}^{(n)}/(1-\rho_2^{(n+1)})$
  $\boldsymbol{w}^{(n)} = (\boldsymbol{v}^{(n+1)} + \epsilon)^{\circ -\frac{1}{2}} \odot (\boldsymbol{m}^{(n+1)})$
  $\boldsymbol{t} \leftarrow \boldsymbol{D}^{\intercal}(\boldsymbol{w}^{(n)} + \boldsymbol{\Delta}_{\boldsymbol{z}}^{\intercal}\boldsymbol{y}_1^{(n)} + \boldsymbol{y}_2^{(n)})$
  $\boldsymbol{x}^{(n+1)} = \mathfrak{G}_{\lambda\|\cdot\|_1}(\boldsymbol{x}^{(n)} - \gamma_1 \boldsymbol{t}, \gamma_1^{\frac{1}{2}})$
  $\boldsymbol{q}^{(n+1)} = \boldsymbol{D}\boldsymbol{x}^{(n+1)}$
  $\boldsymbol{u} \leftarrow 2\boldsymbol{q}^{(n+1)} - \boldsymbol{q}^{(n)}$
  $\boldsymbol{y}_1^{(n)} \leftarrow \boldsymbol{y}_1^{(n)} + \gamma_2 \boldsymbol{\Delta}_{\boldsymbol{z}} \boldsymbol{u}$
  $\boldsymbol{y}_2^{(n)} \leftarrow \boldsymbol{y}_2^{(n)} + \gamma_2 \boldsymbol{u}$
  $\boldsymbol{y}_1^{(n+1)} = \boldsymbol{y}_1^{(n)} - \gamma_2 \mathfrak{G}_{\eta\|\cdot\|_1}(\gamma_2^{-1}\boldsymbol{y}_1^{(n)}, \gamma_2^{-\frac{1}{2}})$
  $\boldsymbol{y}_2^{(n+1)} = \boldsymbol{y}_2^{(n)} - \gamma_2 \mathcal{P}_{[a,b]^N}(\gamma_2^{-1}\boldsymbol{y}_2^{(n)})$
  $n \leftarrow n+1$
end while

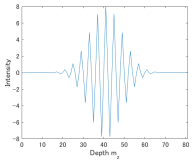


Fig. 2. Discrete model of the coherence function. Amplitude: $\alpha_p = 8$, standard deviation: $\sigma_p = 8$, and angular frequency: $\omega_p = 0.25\pi$.

2) Restoration performance: We simulate the reconstruction of artificial volumetic data to evaluate the processing time of the proposed method and compare it with that of the conventional one. The experimental environment in this section is shown in Table I. Fig. 3 shows the artificially generated volumetric data. The result of the restoration when MSE reaches to 2.00×10^{-5} is shown in Fig. 4, and the result when MSE reaches to 1.80×10^{-5} is shown in Fig. 5. The variation of MSE with respect to the running time of the algorithm is shown in Fig. 7. The time required to reach a certain MSE is summarized in Tables II and III. For this OCT data restoration, we used $\gamma_1 = 1.00 \times 10^{-3}$, $\gamma_2 = 476.1905$, $\epsilon = 1.00 \times 10^{-8}$, and UDHT [12] of level 1 as the synthesis dictionary $\boldsymbol{D}$.

From Tables II and III, it is confirmed that the restoration time is reduced when the acceleration method is introduced. From Figs. 4-6, it is seen that the restoration performance of the proposed acceleration is almost the same as that of the original in Algorithm 3. In particular, the processing time

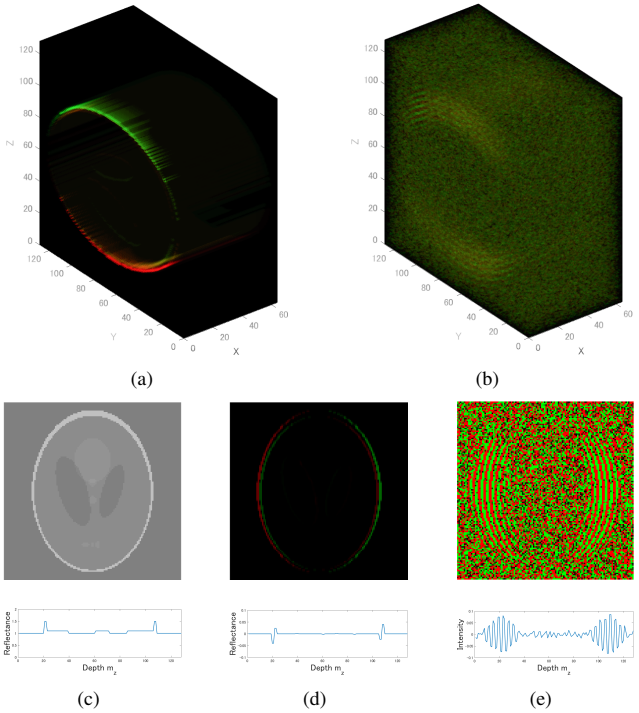


Fig. 3. Example of artificial volumetric data. (a) Reflectance distribution: $\boldsymbol{r}$ of size $64 \times 128 \times 128$ voxels. The function Phantom in MATLAB R2020b was used with the option 'Modified Shepp-Logan', rescaled between luminance $a = 1.00, b = 1.50$, volumized by iterating Y-Z slices in the X direction, and converted to reflectance. (b) Observation signal $\boldsymbol{v}$. $\boldsymbol{P}$ is set to the convolution with the function in Fig. 2. $\boldsymbol{w}$ is set to AWGN with zero mean and standard deviation 2.0×10^{-1}. The green and red voxels mean positive and negative values, respectively, but the brightness of (a) and (b) are increased for better visibility. Y-Z slice at the center of X (middle). Signal sequence in the Z direction at the Y-Z center (bottom). (c) Reflectance distribution: $\boldsymbol{r}$. (d) Reflectance distribution $\boldsymbol{r}$. (e) Observation signal $\boldsymbol{v}$.

TABLE II
COMPARISON OF EXECUTION TIME TO REACH MSE $= 2.00 \times 10^{-5}$

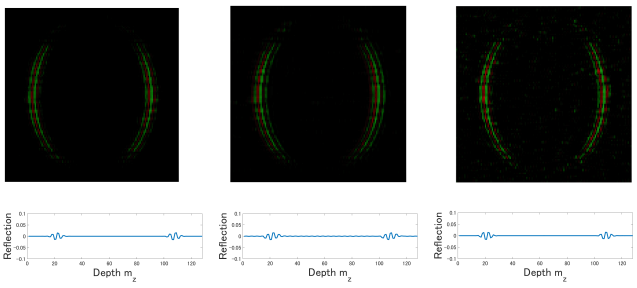
	Conventional	RMSPropGraves	Adam
Time [sec]	209.1	55.2	135.2
Speed (times)	1	3.79	1.55

with RMSpropGraves was reduced to about $1/4$ for MSE of 2.00×10^{-5} and about $1/7$ for MSE of 1.80×10^{-5}.

B. Evaluation of fixed-point arithmetic

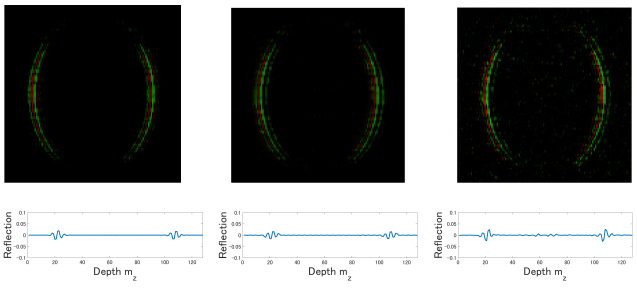
In this section, we conduct 2-D color image restoration simulation based on the TV regularization [16] using PDS to measure the processing time to reach a certain peak signal-to-noise ratio (PSNR) and evaluate the restoration performance. Fig. 8 shows the processing flow of this evaluation. The original observation image and PDS parameters are represented in double precision floating point number type. However, PDS iterations are performed by casting them into the following three types of representations.

- Double precision floating point type (hereafter double)
- Single precision floating point type (hereafter single)
- 32-bit signed integer (hereinafter int32)



(a) (b) (c)

Fig. 4. Reconstruction results at MSE $= 2.00 \times 10^{-5}$. Y-Z slice at the center of X (top). Signal sequence in the Z direction at the center of Y-Z (bottom). (a) Conventional method, computation time : 209.1 sec (b) Method with RMSpropGraves, computation time : 55.2 sec (c) Method with Adam, computation time : 135.2 sec. However, the brightness is increased for better readability.



(a) (b) (c)

Fig. 5. Reconstruction results at MSE $= 1.80 \times 10^{-5}$. Y-Z slice at the center of X (top). Signal sequence in the Z direction at the center of Y-Z (bottom). (a) Conventional method, computation time : 511.3 sec (b) Method with RMSpropGraves, computation time : 74.4 sec (c) Method with Adam, computation time : 769.2 sec

1) Problem setting: Let us consider the following problem setting.

$$\hat{\boldsymbol{s}} = \underset{\boldsymbol{s}\in\mathbb{R}^L}{\arg\min} \frac{1}{2\lambda}\|\boldsymbol{v} - \boldsymbol{P}\boldsymbol{s}\|_2^2 + \iota_{[0,1]^N}(\boldsymbol{s}) + \|\boldsymbol{D}\boldsymbol{s}\|_{1,2}, \quad \text{s.t. } \boldsymbol{D}\boldsymbol{s} \in [0,1]^N. \quad (14)$$

In order to solve this problem, we can adopt Algorithm 6, which is based PDS, where we let

$$f(\boldsymbol{x}) = F(\boldsymbol{D}\boldsymbol{x}) = \frac{1}{2\lambda}\|\boldsymbol{v} - \boldsymbol{P}\boldsymbol{x}\|_2^2, \quad (15)$$

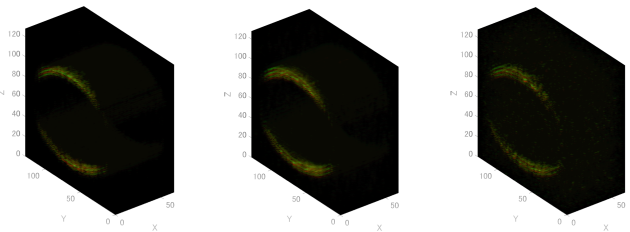
$$g(\boldsymbol{x}) = \|\boldsymbol{D}\boldsymbol{x}\|_{1,2}, \quad (16)$$

$$h(\boldsymbol{G}\boldsymbol{x}) = \iota_{[0,1]^N}(\boldsymbol{x}). \quad (17)$$

In Algorithm 6, $\mathcal{M}_{\gamma^{-1}\|\cdot\|_{1,2}}(\gamma^{-1}\boldsymbol{x})$ is defined by

$$\left[\mathcal{M}_{\gamma^{-1}\|\cdot\|_{1,2}}(\gamma^{-1}\boldsymbol{x})\right]_n \triangleq [\boldsymbol{x}]_n \odot \max\{1 - \|[\boldsymbol{x}]_n\|_2^{-1}, 0\}. \quad (18)$$

2) Restoration performance: We restore 24 images from Kodak Lossless Ture Color Image set by the TV regularization [16], and compare and evaluate the recovery accuracy and processing time of double, single and int32. In this study, we use GPU for the iterative computation of PDS. The experimental environment in this section is shown in Table V.



(a) (b) (c)

Fig. 6. Reconstruction results at MSE $= 1.80 \times 10^{-5}$. (a) Original method (b) RMSpropGraves (c) Adam, where the brightness is increased for better readability.

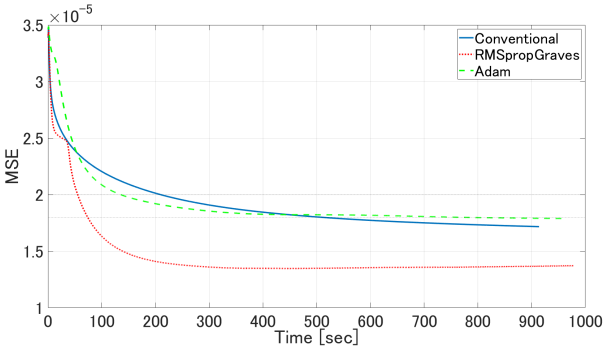


Fig. 7. Variation of MSE versus runtime in artificial volumetric data recovery for the original method (solid blue line), RMSPropGraves (dotted red line), and Adam (dashed green line).

Fig. 9 shows three examples of original images, which are to be artificially degraded by pixel loss process with missing ratio of 50% and AWGN with zero mean and standard deviation of 0.01. The observed images are shown in Fig. 10. Fig. 11 shows the restored results with a run time of 2.0 seconds, where Algorithm 9 with $\lambda = 0.01, \gamma_1 = 0.80$ and $\gamma_2 = 0.1563$ is used, and $\boldsymbol{D}$ denotes the matrix for taking differences between adjacent pixels in the horizontal and vertical direction. The average PSNR of 24 images for the execution time is shown in Fig. 12, and the PSNR for the execution time of 2.0 sec is shown in Table VI. Figs. 11, 12 and Table VI show that single shows the best PSNR at 2.0 sec, followed by int32. All of double, single, and int32 converge to almost the same PSNR. There is no significant difference in the recovery accuracy between floating-point and fixed-point types. On the other hand, there is significant difference in the processing time of double from the others.

V. Conclusions

In this study, we proposed to introduce acceleration method and fixed-point arithmetic implementation to speed up high-dimensional signal restoration using PDS. For the acceleration method, we conducted restoration on artificially generated volumetric data based on the OCT observation model. The proposed method was able to process the data in a shorter time than the conventional one. As for the fixed-point arithmetic,

TABLE III
COMPARISON OF EXECUTION TIME TO REACH MSE $= 1.80 \times 10^{-5}$

	Conventional	RMSPropGraves	Adam
Time [sec]	511.3	74.4	769.2
Speed (times)	1	6.87	0.66

TABLE IV
SIMULATION PARAMETERS FOR OCT DATA RESTORATION

	Conventional	RMSPropGraves	Adam
λ	2.62×10^{-2}	1.68	1.05×10^{-1}
η	8.19×10^{-1}	2.56×10^{-2}	4.10×10^{-1}
ρ_1	-	9.50	0.92
ρ_2	-	-	0.995

we conducted restoration of 2-D color images based on the TV regularization. From the experiments, we found that the fixed-point type can recover the 2-D color image with almost the same accuracy as the floating-point type. Regarding to the processing speed, single showed better results than int32 on GPU. The GPU implementation used in this study did not show substantial advantages in fixed-point arithmetic over single-precision floating-point arithmetic. Unlike GPUs, FPGAs and embedded CPUs can be expected to perform fixed-point arithmetic in a shorter time than floating-point one. In the future, we will implement fixed-point arithmetic in FPGAs and embedded CPUs to achieve faster restoration that is more flexible and suitable for edge computation.

ACKNOWLEDGMENT

This work was supported under the framework of international cooperation program managed by the National Research Foundation of Korea (NRF-2020K2A9A2A08000177) and Japan Society for the Promotion of Science (JSPS-JPJSBP120208804), and supported by JSPS KAKENHI Grant Number JP19H04135 and JP19H02151.

APPENDIX A
PMSPROPGRAVES

The algorithm of RMSpropGraves [8] is shown in Algorithm 7. RMSpropGraves stores the sum of squares of the gradients in each dimension and calculates the square root for each element to account for the amount of gradient update in each dimension. In addition, it is corrected to suppress the influence of the most recently acquired gradient while exponentially attenuating the previous gradients, so that it is less strongly influenced by the most recently acquired gradient information while generally ignoring the gradient information in the distant past.

APPENDIX B
ADAM

The algorithm of Adam [9] is shown in Algorithm 8. Like RMSPropGraves, Adam also considers the gradient update for each dimension. Adam is a combination of the momentum method and RMSprop. ρ_1, ρ_2 are an exponential decay parameter for moment estimation, often set to around $\rho_1 = 0.9$, $\rho_2 = 0.99$ in Adam.

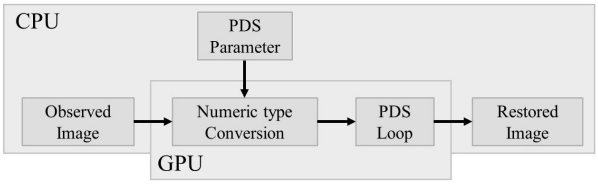


Fig. 8. The signal flow of our simulation of restoration with TV regularization

Algorithm 6 PDS for (14)

Input: $\boldsymbol{x}^{(0)}, \lambda > 0, \gamma_1 > 0, \gamma_2 > 0, n \leftarrow 0$
Output: $\boldsymbol{x}^{(n)}$

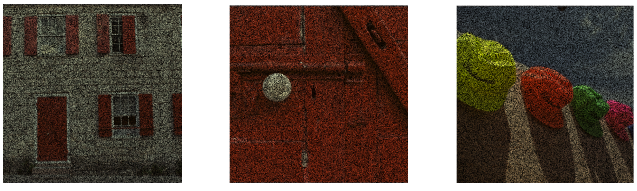
$\boldsymbol{y}^{(0)} = \boldsymbol{D}\boldsymbol{x}^{(0)}$
while A stopping criterion is not satisfied **do**
  $\boldsymbol{x}^{(n+1)} = \mathcal{P}_{[0,1]^N}(\boldsymbol{x}^{(n)} - \gamma_1(\boldsymbol{P}^{\mathsf{T}}(\frac{1}{\lambda}\boldsymbol{P}(\boldsymbol{x}^{(n)}) - \boldsymbol{v}) + \boldsymbol{D}^{\mathsf{T}}\boldsymbol{y}^{(n)}))$
  $\boldsymbol{y}^{(n)} = \boldsymbol{y}^{(n)} + \gamma_2(2\boldsymbol{x}^{(n+1)} - \boldsymbol{x}^{(n)})$
  $\boldsymbol{y}^{(n+1)} = \boldsymbol{y}^{(n)} - \mathcal{M}_{\gamma_2^{-1}\|\cdot\|_{1,2}}(\gamma_2^{-1}\boldsymbol{y}^{(n)})$
  $n \leftarrow n + 1$
end while

TABLE V
SPECIFICATIONS OF SIMULATIONS FOR 2-D COLOR IMAGE RESTORATION

Simulation Tools	MATLAB R2021a
OS	Windows10 pro
CPU	Intel Core i7-9800X
GPU	NVIDIA Quadro GV100
Memory	32GB

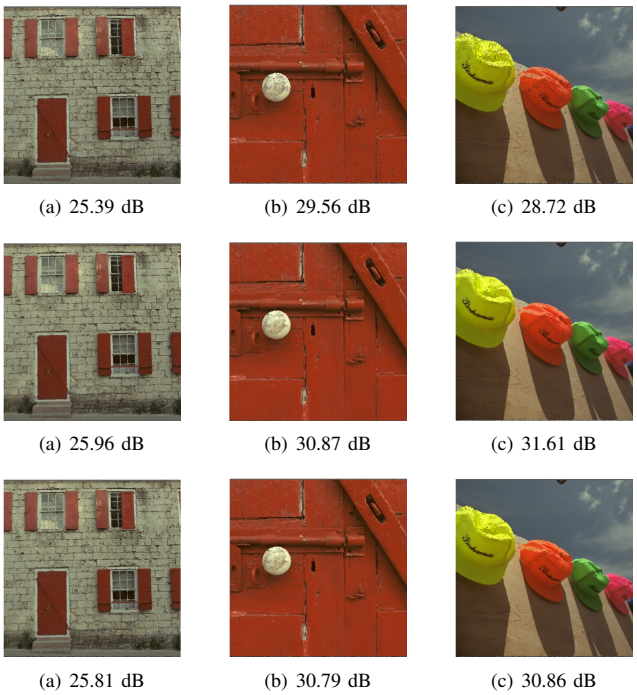
(a) kodim01 (b) kodim02 (c) kodim03

Fig. 9. Three examples of original images of size 512×512 pixels in Kodak set.



(a) 9.91 dB (b) 11.66 dB (c) 10.95 dB

Fig. 10. Observed images through pixel loss with missing ratio of 50% and AWGN with zero mean and standard deviation 0.01. The PSNRs are shown.



(a) 25.39 dB (b) 29.56 dB (c) 28.72 dB

(a) 25.96 dB (b) 30.87 dB (c) 31.61 dB

(a) 25.81 dB (b) 30.79 dB (c) 30.86 dB

Fig. 11. Restored images with of double (top), single (middle) and int32 (bottom) arithmetic. PSNR at time 2.0 sec is also shown.

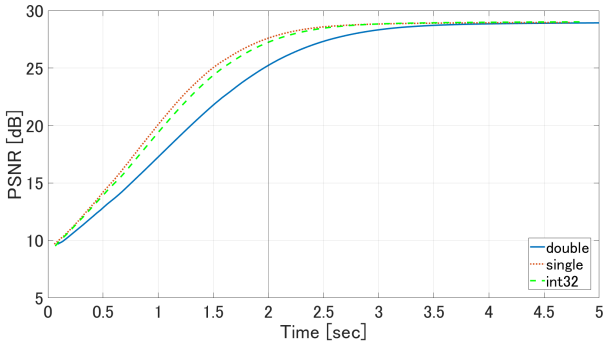


Fig. 12. The average PSNR for the execution time of 24 images. Solid line: double, dashed line: single, and dotted line: int32.

Algorithm 7 RMSpropGraves Algorithm

Input: $\boldsymbol{x}^{(0)}, \boldsymbol{m}^{(0)}, \boldsymbol{v}^{(0)}, \gamma > 0, \rho_1 > 0, \epsilon > 0, n \leftarrow 0$
Output: $\boldsymbol{x}^{(n)}$

while A stopping criterion is not satisfied **do**
  $\boldsymbol{g}^{(n)} = \nabla f(\boldsymbol{x}^{(n)})$
  $\boldsymbol{m}^{(n+1)} = \rho_1 \boldsymbol{m}^{(n)} + (1-\rho_1)\boldsymbol{g}^{(n)}$
  $\boldsymbol{v}^{(n+1)} = \rho_1 \boldsymbol{v}^{(n)} + (1-\rho_1)(\boldsymbol{g}^{(n)})^{\circ 2}$
  $\Delta \boldsymbol{x}^{(n)} = (\boldsymbol{v}^{(n+1)} - (\boldsymbol{m}^{(n+1)})^{\circ 2} + \epsilon)^{\circ -\frac{1}{2}} \odot \boldsymbol{g}^{(n)}$
  $\boldsymbol{x}^{(n+1)} = \boldsymbol{x}^{(n)} - \gamma \Delta \boldsymbol{x}^{(n)}$
  $n \leftarrow n+1$
end while

TABLE VI
PSNR FOR THE EXECUTION TIME OF 2.0 SEC

	double	single	int32
kodim01	25.39	**25.96**	25.81
kodim02	29.56	**30.87**	30.79
kodim03	28.72	**31.61**	30.86
kodim04	29.35	**31.76**	31.48
kodim05	23.35	**24.59**	24.43
kodim06	22.30	**25.59**	25.00
kodim07	28.87	**30.74**	30.42
kodim08	19.27	**22.09**	21.75
kodim09	26.85	**29.83**	29.57
kodim10	27.20	**29.46**	29.18
kodim11	26.96	**27.63**	27.47
kodim12	22.94	**28.29**	27.18
kodim13	21.80	**23.12**	22.92
kodim14	25.31	**27.13**	26.66
kodim15	25.12	27.92	**27.94**
kodim16	27.59	**29.79**	29.44
kodim17	29.12	**30.13**	30.06
kodim18	26.20	**27.02**	26.89
kodim19	23.26	**25.68**	25.39
kodim20	17.55	**23.19**	21.98
kodim21	25.58	**27.12**	26.92
kodim22	25.87	**28.31**	27.92
kodim23	24.52	27.62	**28.09**
kodim24	23.86	**25.44**	25.32

Algorithm 8 Adam Algorithm

Input: $\boldsymbol{x}^{(0)}, \boldsymbol{m}^{(0)}, \boldsymbol{v}^{(0)}, \gamma > 0, \rho_1 > 0, \rho_2 > 0, \epsilon > 0, n \leftarrow 0$
Output: $\boldsymbol{x}^{(n)}$

while A stopping criterion is not satisfied **do**
  $\boldsymbol{g}^{(n)} = \nabla f(\boldsymbol{x}^{(n)})$
  $\boldsymbol{m}^{(n+1)} = \rho_1 \boldsymbol{m}^{(n)} + (1-\rho_1)\boldsymbol{g}^{(n)}$
  $\boldsymbol{v}^{(n+1)} = \rho_2 \boldsymbol{v}^{(n)} + (1-\rho_2)(\boldsymbol{g}^{(n)})^{\circ 2}$
  $\hat{\boldsymbol{m}}^{(n)} = \boldsymbol{m}^{(n+1)} / (1-\rho_1^{(n+1)})$
  $\hat{\boldsymbol{v}}^{(n)} = \boldsymbol{v}^{(n+1)} / (1-\rho_2^{(n+1)})$
  $\Delta \boldsymbol{x}^{(n)} = (\hat{\boldsymbol{v}}^{(n)} + \epsilon)^{\circ -\frac{1}{2}} \odot \hat{\boldsymbol{m}}^{(n)}$
  $\boldsymbol{x}^{(n+1)} = \boldsymbol{x}^{(n)} - \gamma \Delta \boldsymbol{x}^{(n)}$
  $n \leftarrow n+1$
end while

REFERENCES

[1] I. N. Bankman, "Handbook of Medical Image Processing and Analysis," *Handbook of Medical Image Processing and Analysis*, 2009.

[2] N. Komodakis and J. C. Pesquet, "Playing with duality: An overview of recent primal-dual approaches for solving large-scale optimization problems," *IEEE Signal Processing Magazine*, vol. 32, no. 6, pp. 31–54, 11 2015.

[3] A. Chambolle and T. Pock, "A First-Order Primal-Dual Algorithm for Convex Problems with Applications to Imaging," *Journal of Mathematical Imaging and Vision 2010 40:1*, vol. 40, no. 1, pp. 120–145, 12 2010.

[4] L. Condat, "A primal-dual splitting method for convex optimization involving Lipschitzian, proximable and linear composite terms," *Journal of Optimization Theory and Applications*, 2013.

[5] P. L. Combettes and J.-C. Pesquet, "Primal-dual splitting algorithm for solving inclusions with mixtures of composite, Lipschitzian, and parallel-sum monotone operators," *Set-Valued and Variational Analysis*, vol. 20, no. 2, pp. 307–330, 6 2011.

[6] B. C. Vũ, "A splitting algorithm for dual monotone inclusions involving cocoercive operators," *Advances in Computational Mathematics 2011 38:3*, vol. 38, no. 3, pp. 667–681, 11 2011.

[7] G. Fujii, Y. Yoshida, S. Muramatsu, S. Ono, S. Choi, T. Ota, F. Nin, and H. Hibino, "OCT Volumetric Data Restoration with Latent Distribution of Refractive Index," *Proceedings - International Conference on Image Processing, ICIP*, vol. 2019-September, pp. 764–768, 9 2019.

[8] A. Graves, "Generating Sequences With Recurrent Neural Networks," Tech. Rep., 2013.

[9] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, 12 2014.

[10] K. R. Rao and P. C. Yip, "The transform and data compression handbook," p. 388, 2001.

[11] D. S. Taubman and M. W. Marcellin, "JPEG2000 Image Compression Fundamentals, Standards and Practice," *JPEG2000 Image Compression Fundamentals, Standards and Practice*, 2002.

[12] M. Unser, "Texture Classification and Segmentation Using Wavelet Frames," *IEEE Transactions on Image Processing*, vol. 4, no. 11, pp. 1549–1560, 1995.

[13] M. Elad, M. A. Figueiredo, and Y. Ma, "On the role of sparse and redundant representations in image processing," *Proceedings of the IEEE*, vol. 98, no. 6, pp. 972–982, 2010.

[14] M. Elad, "Sparse and redundant representations: From theory to applications in signal and image processing," *Sparse and Redundant Representations: From Theory to Applications in Signal and Image Processing*, pp. 1–376, 2010.

[15] S. Ono, "Primal-Dual Plug-and-Play Image Restoration," *IEEE Signal Processing Letters*, vol. 24, no. 8, pp. 1108–1112, 8 2017.

[16] L. I. Rudin, S. Osher, and E. Fatemi, "Nonlinear total variation based noise removal algorithms," *Physica D: Nonlinear Phenomena*, vol. 60, no. 1-4, pp. 259–268, 11 1992.

[17] S. Muramatsu, S. Chai, S. Ono, T. Ota, F. Nin, and H. Hibino, "Oct Volumetric Data Restoration via Primal-Dual Plug-and-Play Method," *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, vol. 2018-April, pp. 801–805, 9 2018.