APSIPA Transactions on Signal and Information Processing, 2025, 14, e3

Original Paper

RTL Evaluation of ℓ_2-Norm Approximation with Rotated ℓ_1-Norm for 2-Tuple Arrays

Shu Abe[1], Yuya Kodama[1], Hiroyoshi Yamada[2] and Shogo Muramatsu[2*]

[1] *Graduate School of Science & Technology, Niigata University, Japan*
[2] *Faculty of Engineering, Niigata University, Japan*

ABSTRACT

This study proposes a high-precision fast approximation method for the ℓ_2-norm evaluation of 2-tuple data arrays using a rotated ℓ_1-norm evaluation with fixed-point arithmetic. In several signal processing applications, such as image restoration with isotropic total variation (TV) and one with complex ℓ_1-norm regularization, a large number of calculations for the 2-tuple ℓ_2-norm are frequently required. To achieve a hardware (HW)-friendly calculation, the square and square root operations involved in the ℓ_2-norm calculation should be adequately approximated. However, several existing techniques have been challenged with respect to approximations. Thus, in this paper, a HW-friendly approximation algorithm is proposed. The proposed method uses the fact that the upper bound of the surface of a first-order rotational cone traces a second-order cone, that is, the ℓ_2-cone. As a result, less variable multiplication is required, and parallel implementation is easily achieved using fixed-point arithmetic. To demonstrate the effectiveness of the proposed method, it was applied to image restoration, and then its performance on field programmable gate arrays (FPGA) is evaluated in terms of the quality, circuit area, latency, and throughput.

*Corresponding author: Shogo Muramatsu, shogo@eng.niigata-u.ac.jp. This work was supported by JSPS KAKENHI Grant Numbers JP22H00512, JP17H03261, JP19H04135, JP20H02147, and AMD University Program.

Received 02 September 2024; revised 26 December 2024; accepted 28 December 2024
ISSN 2048-7703; DOI 10.1561/116.20240068

The effectiveness of the proposed method is verified by comparing it with typical implementations using commercial circuits.

Keywords: Sparse representations, Fixed point implementation, Soft-thresholding, Embedded vision, FPGA

1 Introduction

The Internet of Things (IoT) plays a significant role in the development of the information society infrastructure [27, 7]. IoT refers to the concept of Internet applications in which many things (devices) are connected through the Internet and physical data are collected by servers. Examples of IoT applications include smart homes, health monitoring, agricultural support, life support, and driving support. These examples are built based on a sensor network that connects a variety of things to the Internet. Because the network processes a large amount of traffic and the server needs to manage a huge amount of general-purpose data within a limited response time, it is necessary to reduce the communication and computation load of the entire system [41, 31, 30, 1].

IoT applications that handle large amounts of data, such as driver assistance technology with cameras and radar to detect require low-power and high-speed systems owing to their nature [16, 12, 15, 40, 20]. As well, medical applications require high performance computing systems for processing large amount of high quality images and high dimensional data [19, 26]. These issues are expected to be solved through hardware (HW) implementation. For example, Xilinx's Alveo, an adaptive accelerator card that accelerates data center workloads, provides high-performance, low-latency, power-efficient computation acceleration for media processing, e.g. deep neural network (DNN)[1] [18]. In IoT applications and medical applications, image acquisition is often performed in severe environments; therefore, image restoration techniques are indispensable. These factors call for image restoration techniques suitable for HW implementation.

Recently, many image restoration methods have utilized the power of sparse modeling. Sparse modeling is a technique that provides a mathematical representation of prior knowledge available for optimization, under the assumption that the essence of the target unknown data can be represented sparsely in some mapped domain. Typical examples to promote sparse representation of images include TV regularization and ℓ_1-norm regularization [38, 35, 29, 8,

[1] Accelerating DNNs with Xilinx Alveo Accelerator Cards can be viewed at https://docs.amd.com/v/u/en-US/wp504-accel-dnns (Oct 2018).

6, 37, 4, 10, 17, 11]. TV can be solved using primal-dual splitting (PDS) [28, 24]. In the HW implementation for high speed processing, we must consider the arithmetic operations in the PDS. However, certain cases require operations that consume circuit resources, for example, variable multiplications and non-linear operations. Image restoration using isotropic TV regularization demands a 2-tuple ℓ_2-norm evaluation. The ℓ_2-norm evaluation requires square root and square operations. These minor expenses results in costs that cannot be ignored, particularly when large amounts of data are involved.

In HW, parallel processing, constant scaling, and fixed-point arithmetic are preferred over serial processing, variable multiplication, and floating-point arithmetic [14, 13, 36]. Let us illustrate examples of partly adopting some of the above strategies. An approach for realizing square root arithmetic is the piecewise constant approximation (PCA). Another approach is coordinate rotation digital computer (CORDIC) [23, 2]. However, these approaches do not allow constant scaling, owing to the calculation of the square operations inside the square root for the ℓ_2-norm computation.

In this study, we focus on the approximation of the ℓ_2-norm for 2-tuple arrays, because it appears in sparsity-promoting solvers with the isotropic TV regularization for image restoration and those with complex ℓ_1-norm regularization. We propose an algorithm for approximating a 2-tuple ℓ_2-norm evaluation. The main contributions of this study are summarized as follows:

- The proposed algorithm reduces the ℓ_2-norm evaluation to simple calculations such as constant scaling, where we introduce the fact that the upper bound of a rotated first-order, or ℓ_1-cone traces a second-order, or ℓ_2-cone [5].
- The performance assessment of the proposed method is conducted using image restoration simulations and hardware implementation reports.

This study extends and reorganizes our previous conference paper [21] by focusing on the evaluation aspects, where the algorithm is revised, evaluation for FPGA is enhanced, and experiments are improved. In [21], a 2-tuple ℓ_2-norm approximation operation was implemented in hardware using high-level synthesis (HLS). In this paper, we model it in register transfer level (RTL) for HW implementation. This allowed us to optimally customize the circuits. In addition, we leverage commercially available IPs from a vendor to speed up and minimize the usage of resources and conduct detailed evaluation in terms of latency and throughput is newly evaluated.

The remainder of this paper is organized as follows. In Section 2, we describe an algorithm that uses the 2-tuple ℓ_2-norm for image restoration and summarize the problems for embedded implementation. In Section 3, we propose an approximation method for the fixed-point implementation of the 2-tuple ℓ_2-norm operation as a solution to the problem in Section 2. In Section 4, we evaluate the performance of our proposed method and show the

results of the approximation error for floating-point arithmetic, appropriate arithmetic word length for fixed-point arithmetic, and circuit area for the FPGA implementation. In Section 5, we conclude this study and discuss future prospects.

2 Review of Image Restoration

Let us review the problem settings relating to image restoration that use the ℓ_2-norm evaluation of 2-tuple elements. For context setting, we briefly describe the isotropic TV regularization and the complex ℓ_1-norm regularization. In this paper, the term "2-tuple array" refers to an array of 2-tuple elements. Note that we refer to an element in $\mathbb{R}^2$ and $\mathbb{C}$ as a 2-tuple element in common.

2.1 Isotropic TV Regularization

Figure 1(a) shows a sparsity promoting image restoration model with an analysis dictionary $\mathbf{\Delta} \in \mathbb{R}^{n\times m}$. A problem concerning isotropic TV regularization is categorized in the model in Figure 1(a), where $\mathbf{u} \in \mathbb{R}^m$ is an unknown original image and $\mathbf{d} \in \mathbb{R}^n$ is a vector consisting of the first-order differences of $\mathbf{u}$ analyzed through $\mathbf{\Delta}$ in the vertical and horizontal directions, $\mathbf{v} \in \mathbb{R}^q$ is an observation of $\mathbf{u}$ and assumed to be measured through the process $\mathbf{P} \in \mathbb{R}^{q\times m}$ with Additive White Gaussian Noise (AWGN) $\mathbf{w} \in \mathbb{R}^q$. Their relation can be formulated as

$$\mathbf{d} = \mathbf{\Delta}\mathbf{u}, \tag{1}$$

$$\mathbf{v} = \mathbf{P}\mathbf{u} + \mathbf{w}. \tag{2}$$

An image restoration problem based on the isotropic TV regularization is formulated as

$$\hat{\mathbf{u}} = \arg \min_{\mathbf{u}\in[u_{\min},u_{\max}]^n} \frac{1}{2}\|\mathbf{P}\mathbf{u} - \mathbf{v}\|_2^2 + \lambda\|\mathbf{\Delta}\mathbf{u}\|_{1,2}, \tag{3}$$

where $\|\cdot\|_2$ and $\|\cdot\|_{1,2}$ denote the ℓ_2-norm and mixed ℓ_1-ℓ_2-norm, respectively, $u_{\min}$ and $u_{\max}$ are the minimum and maximum values of elements in $\mathbf{u}$, and λ is a regularization parameter [29]. The mixed ℓ_1-ℓ_2-norm is defined by

$$\|\mathbf{x}\|_{1,2} \coloneqq \sum_i \sqrt{\sum_j x_{i,j}^2}, \tag{4}$$

where $x_{i,j}$ denotes the j-th element of the i-th tuple. Note that $\|\mathbf{\Delta}\mathbf{u}\|_{1,2}$ is refered to as the isotropic TV of $\mathbf{u}$.

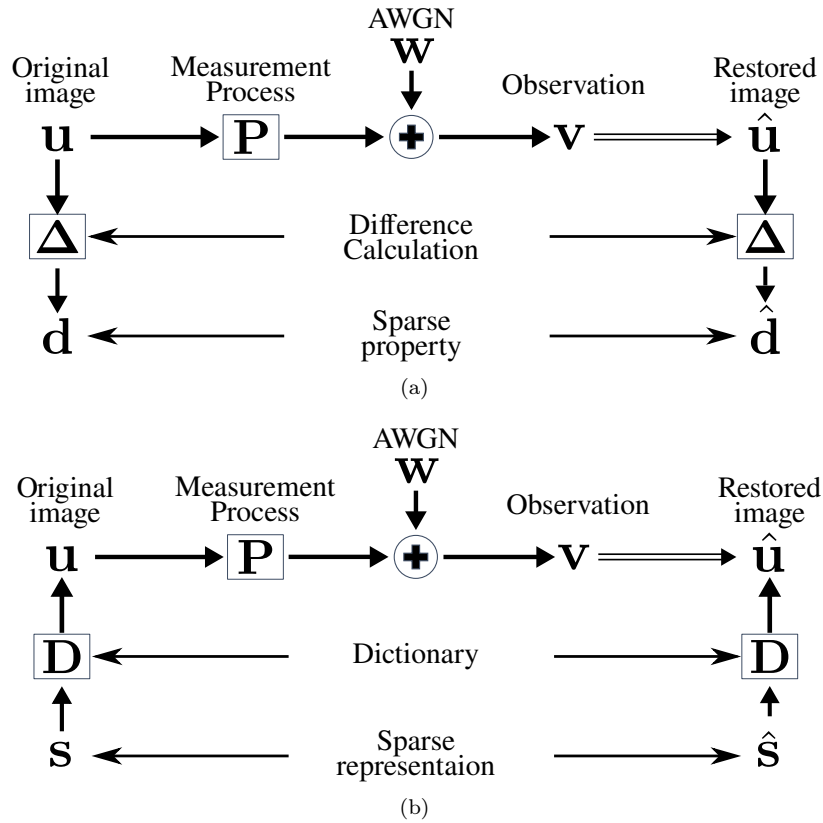


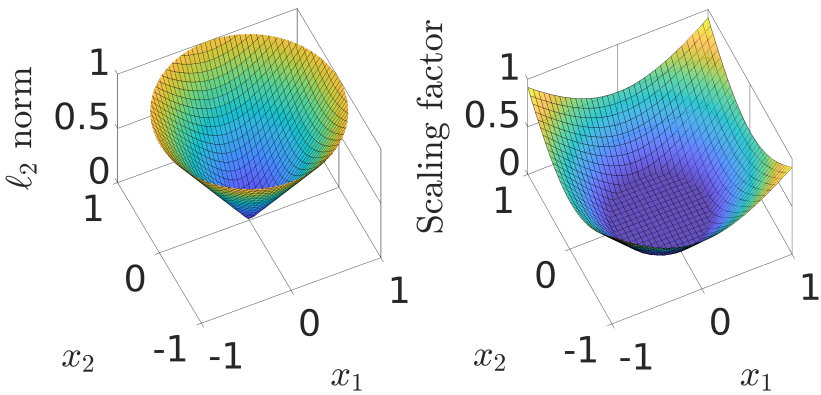
Figure 1: Typical sparsity promoting image restoration models, where (a) is an analysis dictionary model and (b) is a synthesis dictionary model.

We can adopt the PDS algorithm to solve the problem in (3) (see Appendix B for the details) [9, 28]. In the PDS solver, the generalized soft-thresholding expressed by

$$\left[\mathrm{prox}_{\frac{1}{\tau}\|\cdot\|_{1,2}}(\mathbf{x})\right]_i = [\mathbf{x}]_i \odot \max\left(\mathbf{1} - \tau\mathbf{1} \oslash \|[\mathbf{x}]_i\|_2, \mathbf{0}\right), \tag{5}$$

is used as the proximity operator of $\|\cdot\|_{1,2}$, where $[\cdot]_i$ denotes the i-th tuple consisting of the vertical and horizontal differences at the i-th position, $\tau \in [0, \infty)$ denotes the threshold value. The setting of the regularization parameter τ is an important factor that affects the quality of the image restoration, and its optimal value depends on the characteristics of the image and the noise level. $\odot$ and $\oslash$ denote element-wise multiplication and division, respectively, $\max(\cdot, \cdot)$ denotes a vector consisting of greater elements between the first and

second argument elementwisely, **0** and **1** denote the vectors of zeros and ones, respectively. (5) performs the soft-thresholding for every element in the i-th tuple. In the isotropic TV case, each tuple has two elements and consists of differences between adjacent pixels in the vertical and horizontal directions. The proximity operator and (5) are shown in Figure 2(a) and Figure 2(b), where $\tau = 0.5$ respectively.



(a) (b)

Figure 2: (a) The ℓ_2-norm evaluation of $\|(x_1, x_2)^\intercal\|_2$, (b) Generalized soft-thresholding of $\max(1 - \tau/\|(x_1, x_2)^\intercal\|_2, 0)$ in (5), where $\tau = 0.5$.

2.2 Complex ℓ_1-norm Regularization

Figure 1(b) shows a sparsity promoting image restoration model with a synthesis dictionary $\mathbf{D} \in \mathbb{K}^{m\times n}$, where $\mathbb{K}$ is a field such as $\mathbb{R}$ and $\mathbb{C}$. The observation image $\mathbf{v} \in \mathbb{K}^q$ is assumed to be

$$\mathbf{v} = \mathbf{P}\mathbf{u} + \mathbf{w}, \tag{6}$$

where $\mathbf{P} \in \mathbb{K}^{q\times m}$ is the measurement process, $\mathbf{w} \in \mathbb{K}^q$ is AWGN, and $\mathbf{u} \in \mathbb{K}^m$ is an unknown original image represented by

$$\mathbf{u} = \mathbf{D}\mathbf{s}. \tag{7}$$

The least absolute shrinkage and selection operator (Lasso) problem is categorized in the model in Figure 1(b) [10], and the problem setting is formulated as

$$\hat{\mathbf{s}} = \arg\min_{\mathbf{s}\in\mathbb{K}^n} \frac{1}{2}\|\mathbf{P}\mathbf{D}\mathbf{s} - \mathbf{v}\|_2^2 + \lambda\|\mathbf{s}\|_1, \tag{8}$$

where $\|\cdot\|_1$ denotes the ℓ_1-norm. To solve (8), we can adopt iterative shrinkage thresholding algorithm (ISTA).

In this study, we are interested in the complex-valued case when $\mathbf{u} \in \mathbb{C}^m$. In complex ISTA, the complex soft-thresholding expressed by

$$\left[\text{prox}_{\frac{1}{\tau}\|\cdot\|_1}(\mathbf{x})\right]_i = x_i \cdot \max\left(1 - \tau / |x_i|, 0\right) \tag{9}$$

is used as the proximity operator of $\|\cdot\|_1$, where $[\cdot]_i$ denotes the i-th element of its argument and $x_i = [\mathbf{x}]_i \in \mathbb{C}$ [25].

2.3 Identification of Norm Calculations

In the generalized soft-thresholding in (5), it is necessary to calculate the ℓ_2-norm of the 2-tuple element, i.e.,

$$\|[\mathbf{x}]_i\|_2 = \sqrt{x_{i,1}^2 + x_{i,2}^2}. \tag{10}$$

It is also necessary to take the absolute of complex element in the complex soft-thresholding in (9), i.e.,

$$|x_i| = \sqrt{\Re^2(x_i) + \Im^2(x_i)}. \tag{11}$$

The calculations of two formulas are the same in that both of them take the square root of the sum of the squares of two components. Figure 3 illustrates how all tuples should be evaluated in every iteration of the restoration algorithms.

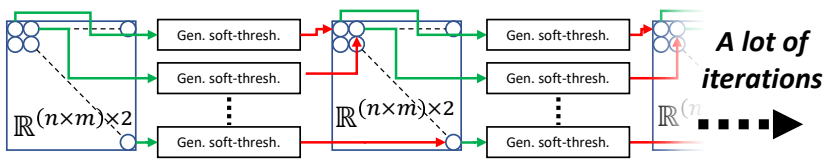


Figure 3: How the soft-thresholding process is applied to each iteration of PDS and ISTA. This diagram illustrates that a number of 2-tuple ℓ_2-norm calculations are required for the process.

2.4 Problems for Embedded Implementation

In an embedded implementation, it is desirable to use fixed-point arithmetic from the viewpoint of circuit resources and calculation speed. As well, constant

scaling is preferable to variable multiplication to reduce circuit resource usage through sophisticated approaches such as canonical signed digit (CSD) representation, reduced adder graph (RAG), and distributed arithmetic (DA) [3, 22, 36]. For the square root function, the calculation can be approximately realized by several existing methods such as piecewise constant approximation (PCA) and CORDIC [33, 34]. The following is an overview of PCA and CORDIC for reference.

2.4.1 PCA

PCA is a method of converting the relationship of the output signals to the input signals into a truth table. This method is equivalent to predefined memoization in programming, which has the advantage of converting complex functions into a simple truth tables. However, when the accuracy of the output signal is increased, the scale of the truth table becomes very large, which increases the amount of resources used in the HW implementation.

2.4.2 CORDIC

CORDIC is an algorithm that can calculate trigonometric functions (sin, cos, tan, arctan, sinh, cosh, tanh, and arctanh), logarithmic functions, exponential functions, and square roots only by shift operations, addition and subtraction operations, and calling predefined constants. Therefore, it has the advantage of low resources usage in HW implementation. However, in the ℓ_2-norm operation, CORDIC can be applied to the square root operation, but not to the square operation. Thus, we cannot avoid to consume circuit resources for the square operation. In addition, parallel processing is not possible because the shift and addition/subtraction operations are performed sequentially.

3 2-tuple ℓ_2-norm Fixed-point Approximation

In this section, we propose a direct approximation method for the 2-tuple ℓ_2-norm evaluation with fixed-point arithmetic to solve the problems discussed in the previous section.

3.1 Second-order Cone via Rotational First-order Cone

Let us first express a key idea of our proposed approximation. We identically represent (10) and (11) in a form without the square root and square operations. We express the real and imaginary part of a complex value x_i as $x_{i,1} = \Re(x_i)$ and $x_{i,2} = \Im(x_i)$, respectively. As a result, the right-hand sides of the two

equations became equal. Thus, in the following section, we discuss the two cases in common.

Our proposed method has the basis of the following theorem:

Theorem 1 (Second-order cone via first-order rotational cone)**.** *Let a rotated first-order cone function* $a_\theta \colon \mathbb{R}^2 \to \mathbb{R}$ *be*

$$a_\theta([\mathbf{x}]_i) \;\coloneqq\; \frac{1}{\sqrt{2}}\Big(|x_{i,1}\cos\theta \;+\; x_{i,2}\sin\theta| \;+\; |x_{i,1}\sin\theta \;-\; x_{i,2}\cos\theta|\Big), \quad (12)$$

where $[\mathbf{x}]_i = (x_{i,1}, x_{i,2})^\intercal \in \mathbb{R}^2$*. Then,* $a_\theta(\cdot)$ *relates to the* ℓ_2*-norm of 2-tuple element* $[\mathbf{x}]_i$ *as*

$$\|[\mathbf{x}]_i\|_2 = \sup_{\theta\in[0,\frac{\pi}{2})} a_\theta([\mathbf{x}]_i). \quad (13)$$

Proof. Let $\angle[\mathbf{x}]_i = \tan^{-1}\frac{x_{i,2}}{x_{i,1}}$. Then,
we see $a_\theta([\mathbf{x}]_i) = \frac{\|[\mathbf{x}]_i\|_2}{\sqrt{2}}\,(|\cos(\angle[\mathbf{x}]_i - \theta)| + |\sin(\angle[\mathbf{x}]_i - \theta)|)$.
When $\theta = \angle[\mathbf{x}]_i - \frac{\pi}{4}$, we have the relation

$$a_{\angle[\mathbf{x}]_i-\frac{\pi}{4}}([\mathbf{x}]_i) = \frac{\|[\mathbf{x}]_i\|_2}{\sqrt{2}}\left(\left|\cos\frac{\pi}{4}\right| + \left|\sin\frac{\pi}{4}\right|\right) = \|[\mathbf{x}]_i\|_2. \quad (14)$$

In order to show $\theta = \angle[\mathbf{x}]_i - \frac{\pi}{4}$ gives the maximum value for $0 \le \angle[\mathbf{x}]_i - \theta < \frac{\pi}{2}$, we can equate the derivative to zero as

$$\frac{\partial a_\theta}{\partial\theta} = \frac{\|[\mathbf{x}]_i\|_2}{\sqrt{2}}\,(-\sin(\angle[\mathbf{x}]_i - \theta) + \cos(\angle[\mathbf{x}]_i - \theta)) = 0. \quad (15)$$

As a result, (13) holds from the fact that $a_\theta(\cdot) = a_{\theta+\frac{\pi}{2}k}(\cdot)$ always holds for any $k \in \mathbb{Z}$. □

3.2 Approximation of 2-tuple ℓ_2-norm Evaluation

Let us derive an approximation method for the ℓ_2-norm evaluation of 2-tuple elements. From Theorem 1, we see that (13) can be approximated by discretizing the angle θ. From this consideration, we propose to approximate 2-tuple ℓ_2-norm as follows:

$$A_N([\mathbf{x}]_i) \;\coloneqq\; \max_{\theta\in\Theta} a_\theta([\mathbf{x}]_i), \Theta \;=\; \{\theta_0, \theta_1, \ldots, \theta_{N-1}\}, \quad \theta_k \;=\; \frac{\pi}{2N}k. \quad (16)$$

Algorithm 1 shows the procedure to realize (16). To visualize the approximation, some plots of (16) are shown in Figure 4. Figure 4(a), (c), and (e) confirm the validity of (16) with $N = 2$, $N = 3$, and $N = 4$, respectively. Figure 4(b), (d), and (f) show the approximation error compared to Figure 2(a).

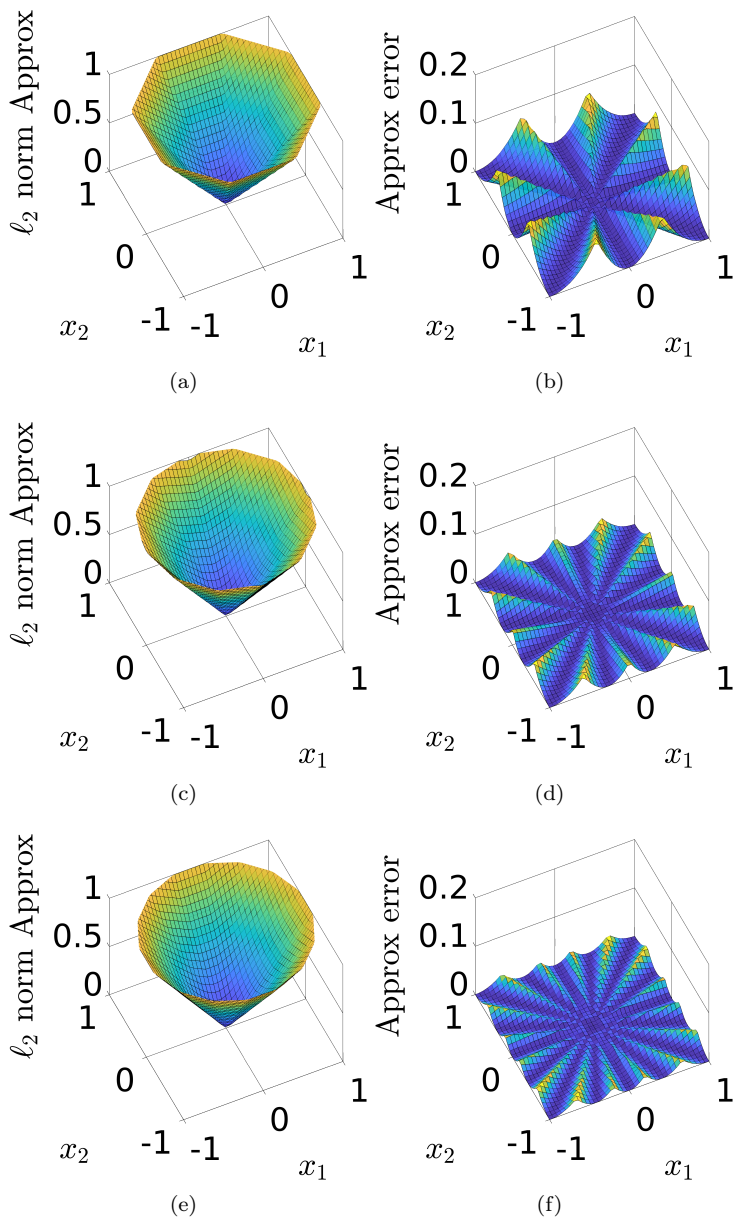


Figure 4: Approximation of ℓ_2-norm evaluation with (16), where (a) $N = 2$, (b) Error of (a), (c) $N = 3$, (d) Error of (c), (e) $N = 4$, and (f) Error of (e).

Algorithm 1 Approximated 2-tuple ℓ_2-norm evaluation.

Input: $(x_1, x_2)^\intercal \in \mathbb{R}^2, \tau > 0, N \in \mathbb{N}$
Output: $a \in \mathbb{R}$
  for each $k \in \{0, 1, 2, \cdots, N-1\}$ **do**
    $c_k, s_k \leftarrow \frac{1}{\sqrt{2}} \cos \frac{\pi}{2N} k, \frac{1}{\sqrt{2}} \sin \frac{\pi}{2N} k$
    $a_k \leftarrow |c_k x_1 + s_k x_2| + |s_k x_1 - c_k x_2|$
  end for each
  $a \leftarrow \max(a_0, a_1, a_2, \ldots, a_{N-1})$

No approximation error is observed in the direction where $\theta = 0$ and $\pi/4$ for Figure 4(b), as indicated by the proof of Theorem 1. The maximum error in the range shown in Figure 4(b) is approximately 0.075541, those shown in Figure 4(d) and (f) are approximately 0.036828 and 0.020862, respectively. One can verify that the error decreases as N increases.

Note that it is possible to calculate $\{c_k\}$ and $\{s_k\}$ in Algorithm 1 in advance and store them as fixed-point constants. In addition, $\{a_k\}$ can be computed independently, allowing parallel processing. Finally, we obtain a simple approximation procedure for the 2-tuple ℓ_2-norm evaluation with fixed-point constant scaling.

3.3 Approximation of Soft-thresholding

Let us derive an approximation method for the generalized soft-thresholding in (5). From Theorem 1 and (16), we propose to approximate (5) as follows:

$$\left[\mathrm{prox}_{\frac{1}{\tau}\|\cdot\|_{1,2}}(\mathbf{x})\right]_i \approx [\mathbf{x}]_i \odot \max\left(\mathbf{1} - \tau \mathbf{1} \oslash A_N([\mathbf{x}]_i), \mathbf{0}\right). \tag{17}$$

Algorithm 2 shows the procedure to realize (17). To visualize the approximation, some plots of (17) are shown in Figure 5. Figure 5(a), (c), and (e) confirm the results of (17) with $N = 2$, $N = 3$, and $N = 4$, where $\tau = 0.5$ respectively. Figure 5(b), (d), and (f) show the approximation error compared to Figure 2(b).

The approximation error results in Figure 5(b), where it is observed that there is no approximation error in the direction where $\theta = 0$ and $\pi/4$, as indicated by Theorem 1. The maximum error in the range shown in Figure 5(b) is approximately 0.041182, those in Figure 5(d) and (f) are approximately 0.017626 and 0.009690, respectively. One can verify that the error decreases as N increases.

As with Section 3.2, note that it is possible to calculate $\{c_k\}$ and $\{s_k\}$ in Algorithm 2 in advance and store them as fixed-point constants. In addition,

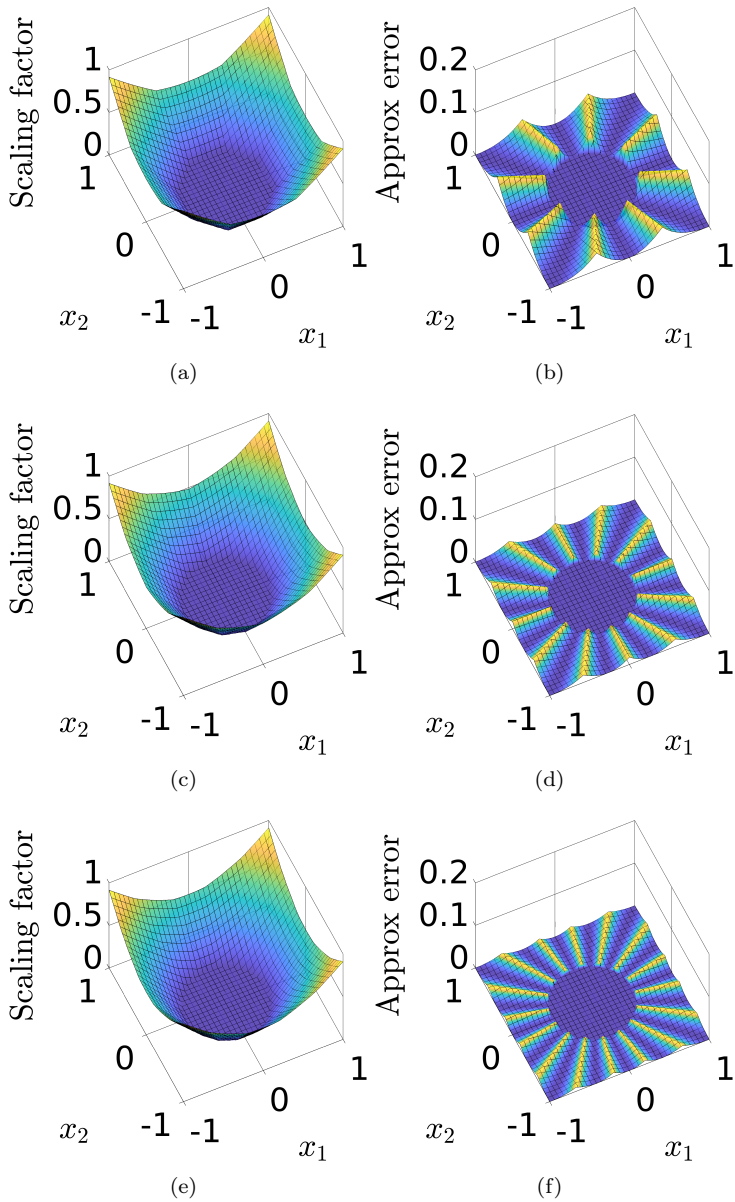


Figure 5: Soft-thresholding function of the generalized soft-thresholding in (17) with $\tau = 0.5$, where (a) $N = 2$, (b) Error of (a), (c) $N = 3$, (d) Error of (c), (e) $N = 4$, and (f) Error of (e).

Algorithm 2 Approximated 2-tuple generalized soft-thresholding.

Input: $(x_1, x_2)^\intercal \in \mathbb{R}^2, \tau > 0, N \in \mathbb{N}$
Output: $(y_1, y_2)^\intercal \in \mathbb{R}^2$
  for each $k \in \{0, 1, 2, \cdots, N-1\}$ **do**
    $c_k, s_k \leftarrow \frac{1}{\sqrt{2}} \cos \frac{\pi}{2N} k, \frac{1}{\sqrt{2}} \sin \frac{\pi}{2N} k$
    $a_k \leftarrow |c_k x_1 + s_k x_2| + |s_k x_1 - c_k x_2|$
  end for each
  $a \leftarrow \max(a_0, a_1, a_2, \ldots, a_{N-1}, \tau)$
  $y_1, y_2 \leftarrow x_1 \left(1 - \frac{\tau}{a}\right), x_2 \left(1 - \frac{\tau}{a}\right)$

$\{a_k\}$ can be computed independently, allowing parallel processing. Finally, we obtain a simple approximation procedure for the 2-tuple generalized soft-thresholding with fixed-point constant scaling.

3.4 Application to Fixed-point Implementations

The evaluation values c_k and s_k of the trigonometric function used in Algorithms 1 and 2 can be constant in advance. As a result, the $\left|c_k x_1 + s_k x_2\right| + \left|s_k x_1 - c_k x_2\right|$ in line 5 of the Algorithms 1 and 2 can be realized by constant multiplication with fixed-point representation.

A comparison of the existing and proposed methods is summarized in Table 1. In contrast to CORDIC and PCA, the proposed method is feasible in terms all of the fixed-point arithmetic, parallel processing and constant scaling.

Table 1: Comparison of existing and proposed methods.

Method	CORDIC	PCA	Proposal
Fixed-point	Feasible	Feasible	Feasible
Parallel Process	Difficult	Feasible	Feasible
Constant scaling	Impossible	Difficult	Feasible

3.5 Analytical Comparison

Let us analyze and compare the complexity of algorithms from an abstract point of view. To give an overview of the performance of the proposed method, the required combinational circuit resources, storage capacities, latency and throughput are formulated as functions of architectural configurations and target device parameters for an abstract performance evaluation. Several

functions are derived for the proposed method, CORDIC without pipeline, CORDIC with pipeline, and PCA. The derived functions are shown in Table 2 (see Appendix A for the derivation).

Table 2: The performance estimation functions for the proposed method, CORDIC without pipeline, CORDIC with pipeline, and PCA, where $\mathcal{A}$, $\mathcal{R}$, $\mathcal{L}$, and $\mathcal{T}$ denote the combinational circuit resources, register capacities, latency, and throughput, respectively.

	Proposed method: $\cdot_{\mathrm{pro}}$	CORDIC without pipeline: $\cdot_{\mathrm{nop}}$
$\mathcal{A}^{(\mathrm{b})}$	$(3N-2)\alpha^{(\mathrm{b})}_{\mathrm{add}} + 2N\alpha^{(\mathrm{b})}_{\mathrm{scaling}}$	$2\alpha^{(\mathrm{b})}_{\mathrm{add}} + 2\alpha^{(\mathrm{b})}_{\mathrm{mult}}$
$\mathcal{R}^{(\mathrm{b})}$	$\rho^{(\mathrm{b})}_{\mathrm{ff}}$	$\rho^{(\mathrm{b})}_{\mathrm{ff}}$
$\mathcal{L}^{(\mathrm{b})}$	$2\xi^{(\mathrm{b})}_{\mathrm{add}} + \xi^{(\mathrm{b})}_{\mathrm{scaling}} + \delta_{\mathrm{ff}}$	$(b+1)\xi^{(\mathrm{b})}_{\mathrm{add}} + \xi^{(\mathrm{b})}_{\mathrm{mult}} + \delta_{\mathrm{ff}}$
$\mathcal{T}^{(\mathrm{b})}$	$2\xi^{(\mathrm{b})}_{\mathrm{add}} + \xi^{(\mathrm{b})}_{\mathrm{scaling}} + \delta_{\mathrm{ff}}$	$(b+1)\xi^{(\mathrm{b})}_{\mathrm{add}} + \xi^{(\mathrm{b})}_{\mathrm{mult}} + \delta_{\mathrm{ff}}$

	CORDIC with pipeline: $\cdot_{\mathrm{pip}}$	PCA: $\cdot_{\mathrm{pca}}$
$\mathcal{A}^{(\mathrm{b})}$	$(b+1)\alpha^{(\mathrm{b})}_{\mathrm{add}} + 2\alpha^{(\mathrm{b})}_{\mathrm{mult}}$	$(n+4)\alpha^{(\mathrm{b})}_{\mathrm{add}} + \alpha^{(\mathrm{b})}_{\mathrm{scaling}} + \alpha^{(\mathrm{b})}_{\mathrm{mult}}$
$\mathcal{R}^{(\mathrm{b})}$	$b\rho^{(\mathrm{b})}_{\mathrm{ff}}$	$\rho^{(\mathrm{b})}_{\mathrm{ff}}$
$\mathcal{L}^{(\mathrm{b})}$	$(b+1)\xi^{(\mathrm{b})}_{\mathrm{add}} + \xi^{(\mathrm{b})}_{\mathrm{mult}} + b\delta_{\mathrm{ff}}$	$(n+3)\xi^{(\mathrm{b})}_{\mathrm{add}} + \xi^{(\mathrm{b})}_{\mathrm{scaling}} + \xi^{(\mathrm{b})}_{\mathrm{mult}} + \delta_{\mathrm{ff}}$
$\mathcal{T}^{(\mathrm{b})}$	$\xi^{(\mathrm{b})}_{\mathrm{add}} + \xi^{(\mathrm{b})}_{\mathrm{mult}} + \delta_{\mathrm{ff}}$	$(n+3)\xi^{(\mathrm{b})}_{\mathrm{add}} + \xi^{(\mathrm{b})}_{\mathrm{scaling}} + \xi^{(\mathrm{b})}_{\mathrm{mult}} + \delta_{\mathrm{ff}}$

In Table 2, $\mathcal{A}$, $\mathcal{R}$, $\mathcal{L}$, and $\mathcal{T}$ denote the combinational circuit resources, storage capacities, latency, and throughput, respectively. The following summarize the meaning of subscripts:

- $\cdot_{\mathrm{pro}}$: Proposed method
- $\cdot_{\mathrm{nop}}$: CORDIC without pipeline
- $\cdot_{\mathrm{pip}}$: CORDIC with pipeline
- $\cdot_{\mathrm{pca}}$: PCA

Since the circuit area and processing time are dependent on the target device and data expressions, we define some meta variables as follows:

- α: Combinational circuit resources
- ρ: Registers (storage resources)
- ξ: Processing time
- δ_{ff}: Setup and hold time of flip-flops (FFs)

The following summarize the meanings of subscripts and superscripts:

- $\cdot_{\mathrm{add}}$: Adder
- $\cdot_{\mathrm{scaling}}$: Constant multiplier

- $\cdot_{\mathrm{mult}}$: Variable multiplier
- $\cdot_{\mathrm{ff}}$: Registers
- $\cdot^{(\mathrm{b})}$: Input bit width b

In Table 2, $\mathcal{A}^{(\mathrm{b})}$ shows that the proposed method is highly dependent on the composite number of ℓ_1-cones N, the CORDIC is mainly dependent on the input bit width b, and the PCA is mainly dependent on the number of nodes n. On the other hand, the result in Table 2 shows that the proposed method is independent of N except for the combinational circuit resources. This means that increasing the accuracy of the proposed method is independent of the time cost.

3.6 Register Transfer Level Modeling

Let us model our proposed 2-tuple ℓ_2-norm approximation operation in RTL for HW implementation, which was built by HLS in the previous work [21]. We propose an RTL model for HW implementation of (16). The RTL modeling is expected to reach a higher performance than that of the previous high-level synthesis approach [21].

4 Performance Evaluation

To evaluate the performance of the proposed method, we first conduct a comprehensive comparison by using abstract equations in Table 2, and then assess the approximation performance of the ℓ_2-norm. Next, we demonstrate the simulation results of image restoration with double precision arithmetic. In addition, we compare the precision of fixed-point arithmetic with those of the existing method. The bit width of the fractional part of the fixed-point implementation is changed, and the mean squared error (MSE) of the output response is compared among the existing and proposed methods. Then, we evaluate each design in terms of the latency, maximum operable frequency, and resource usage of the fixed-point HW implementation and compare the proposed method to other approximation methods. Table 3 summarizes the experimental specifications.

Table 3: Experimental specifications.

Simulation tool	MATLAB R2021b
CPU	Intel(R) Core(TM) i9-10980XE CPU @ 3.00GHz
OS	Linux Ubuntu 20.04.6 LTS

4.1 Abstract Evaluation

Let's compare the performance in terms of the equations shown in Section 3.5 to estimate the HW implementation performance. As an example, we assume the following conditions for meta variables:

- $\alpha^{(\mathrm{b})}_{\mathrm{scaling}} = \beta\alpha^{(\mathrm{b})}_{\mathrm{add}}$, $\alpha^{(\mathrm{b})}_{\mathrm{mult}} = \gamma\alpha^{(\mathrm{b})}_{\mathrm{add}}$
- $\xi^{(\mathrm{b})}_{\mathrm{scaling}} = \beta\xi^{(\mathrm{b})}_{\mathrm{add}}$, $\xi^{(\mathrm{b})}_{\mathrm{mult}} = \gamma\xi^{(\mathrm{b})}_{\mathrm{add}}$,

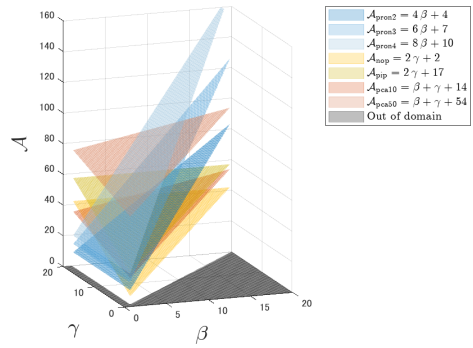
where β and γ are the ratios of HW cost of a constant multiplier and a variable multiplier for an adder, respectively. For convenience, the following will share the ratios of resources and delay time. In general, variable multipliers are more expensive than constant multipliers, and constant multipliers are more expensive than adders. Therefore, we will proceed with the discussion here under the assumption that $1 < \beta < \gamma$. β and γ are variables that depend on the respective HW and cannot be simply defined. Therefore, we evaluate the combinational circuit area, latency, and throughput as functions with variables β and γ. The composite number of ℓ_1-cones for the proposed method was set to $N \in \{2, 3, 4\}$, the number of PCA nodes was set to $n \in \{10, 50\}$, and the input bit width was set to 16. On our assumptions, the following estimations are given by substituting the numbers to the meta variables of equations in Table 2.

Figure 6 shows the evaluation of the combinational circuit area, latency, and throughput with β and γ as variables (see Section 4.5 for the definitions of latency and throughput in this paper). Because of the condition of $1 < \beta < \gamma$, the outside of the domain is shown in gray. For the sake of convenience, $\alpha^{(\mathrm{b})}_{\mathrm{add}}$ and $\xi^{(\mathrm{b})}_{\mathrm{add}}$ are normalized to unit so that it is sufficient to know their relative relationships. The setup and hold time of the register, δ_{ff}, is assumed to be negligible compared to the delay time of the adder, ξ_{add}, and so is ignored.

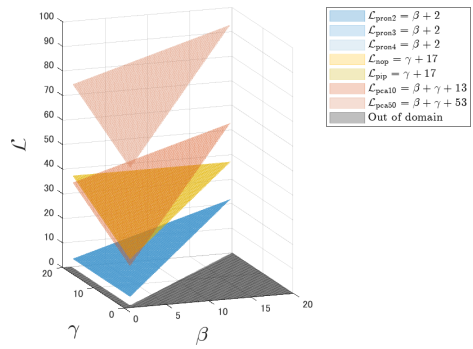
From Figure 6, we see that the proposed method is independent of γ, i.e., the cost of variable multipliers, and the HW performance is influenced mainly by β, i.e., the cost of constant multipliers.

4.2 Evaluation of ℓ_2-norm Approximation

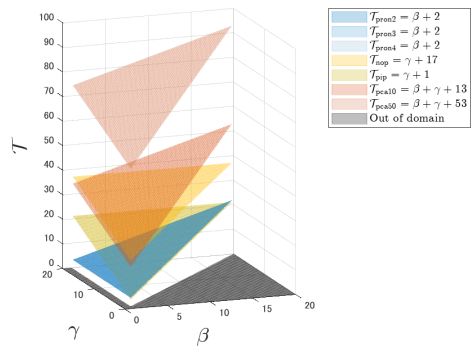
Let us evaluate the arithmetic part of computing the ℓ_2-norm in terms of MSE. We compare the results for all approximation methods (proposed method, CORDIC, and PCA) with the reference in the double precision. The computation and MSE results for each algorithm are shown in Figure 7. CORDIC has the smallest approximation error, followed by the proposed method with $N = 4, N = 3$, and $N = 2$, and PCA. Although the approximation performance of CORDIC is high, the approximation of the proposed method shows



(a)



(b)



(c)

Figure 6: Functions of HW costs with metavariables β and γ, which are the ratios of the costs of a constant multiplier and a variable multiplier for an adder, respectively. On the assumption that $1 < \beta < \gamma$, the outside of the domain is shown in gray. (a) Circuit area, (b) Latency, and (c) Throughput.

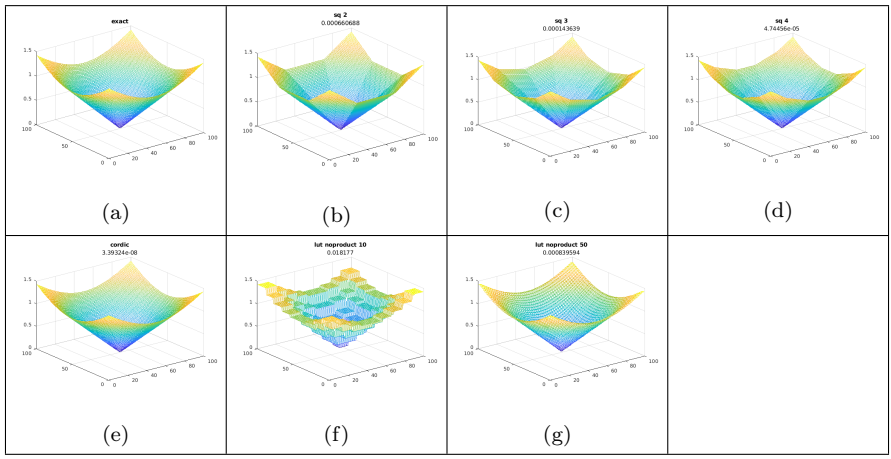


Figure 7: Function plots of the reference and the approximation algorithm, where (a) reference, (b), (c), and (d) the proposed approximation w/ $N = 2$ (MSE: 6.60688×10^{-4}), $N = 3$ (MSE: 1.43639×10^{-4}) and, $N = 4$ (MSE: 4.74456×10^{-5}), respectively. (e) CORDIC (MSE: 3.39324×10^{-8}), (f), (g) PCA w/ $N_p = 10$ (MSE: 0.018177) and $N_p = 50$ (MSE: 8.39594×10^{-4}), respectively.

moderate performance for the image restoration and HW implementation. The details are described in the following sections.

4.3 Image Recovery by TV Regularization

We evaluate the validity of the proposed method through the applications of image restoration. First, we evaluated the TV calculation for restoring a grayscale image. Table 4 summarizes the simulation conditions for the image restoration with the isotropic TV regularization. To find optimal values for the parameters of the image restoration process, the greedy method is commonly used [32]. However, we experimentally searched for the optimal value by manually adjusting the parameters step by step. The choice of τ is important in setting any image restoration problem, and the choice of step sizes γ_1 and γ_2 also have impact for the convergence characteristics of the algorithm. However, since τ, γ_1 and γ_2 do not significantly contribute to the difference between computation process with and without approximation, a simple method was adopted in this study. The simulation was conducted on five images by varying the standard deviation of AWGN. The original five images are shown in Figure 8, which are Cameraman, Barbara, Kodak Image 02, 08 and 19 (Kodim02, Kodim08, and Kodim19).

Let us show the restoration performance of the proposed method with double precision arithmetic. Table 5 shows the MSE for each image and noise

Table 4: Simulation conditions of image restoration modeled by the isotropic TV regularization.

Image size	256×256
Bit depth	8bpp
Scale	$[0, 1]$
Standard deviation of AWGN	$\sigma \in \{5/255, 10/255, 15/255, 20/255\}$
Pixel loss rate	50%
Regularization parameter λ	0.012
Step size parameter γ_1	0.0050
Step size parameter γ_2	37.6984
# of angles	$N \in \{2, 3, 4\}$

Table 5: Simulation results of image restoration by means of the isotropic TV regularization. The experiments are conducted by varying the noise level for the five images in Figure 8. The MSE for the original images of all algorithms are listed. Reference shows the restoration performance without approximation.

Img	Reference	Proposal $N = 2$	Proposal $N = 3$	Proposal $N = 4$	CORDIC	PCA $N_p = 10$	PCA $N_p = 50$
σ =5/255, MSE							
Cameraman	0.001920	0.001938	0.001920	0.001922	0.001920	0.001954	**0.001912**
Barbara	**0.001582**	0.001595	0.001585	0.001584	**0.001582**	0.001729	0.003993
Kodim02	**0.000523**	0.000529	0.000526	0.000524	**0.000523**	0.000543	0.000537
Kodim08	0.004331	0.004345	0.004338	0.004332	0.004331	**0.004147**	0.004434
Kodim19	0.001636	0.001639	0.001642	0.001638	0.001636	**0.001616**	0.009918
σ =10/255, MSE							
Cameraman	0.002175	0.002183	**0.002174**	0.002175	0.002175	0.002438	0.002389
Barbara	0.001912	0.001917	**0.001911**	0.001912	0.001912	0.002100	0.002176
Kodim02	0.000733	**0.000730**	0.000732	0.000733	0.000733	0.001117	0.001085
Kodim08	0.004707	0.004714	0.004710	0.004708	0.004707	**0.004599**	0.005584
Kodim19	0.001915	**0.001912**	0.001917	0.001914	0.001915	0.001992	0.002503
σ =15/255, MSE							
Cameraman	0.002813	0.002808	**0.002807**	0.002810	0.002813	0.003299	0.003090
Barbara	0.002617	0.002613	**0.002612**	0.002614	0.002617	0.002955	0.002781
Kodim02	0.001352	**0.001337**	0.001344	0.001347	0.001352	0.002109	0.002048
Kodim08	0.005461	0.005460	0.005460	0.005459	0.005461	**0.005444**	0.005535
Kodim19	0.002577	0.002566	0.002574	0.002574	0.002577	0.002791	**0.002551**
σ =20/255, MSE							
Cameraman	0.003845	**0.003829**	0.003833	0.003839	0.003845	0.004532	0.004265
Barbara	0.003706	**0.003691**	0.003696	0.003700	0.003706	0.004145	0.003847
Kodim02	0.002402	**0.002376**	0.002388	0.002395	0.002402	0.003502	0.003395
Kodim08	0.006609	**0.006596**	0.006604	0.006605	0.006609	0.006640	0.006612
Kodim19	0.003668	0.003646	0.003658	0.003662	0.003668	0.004053	**0.003554**

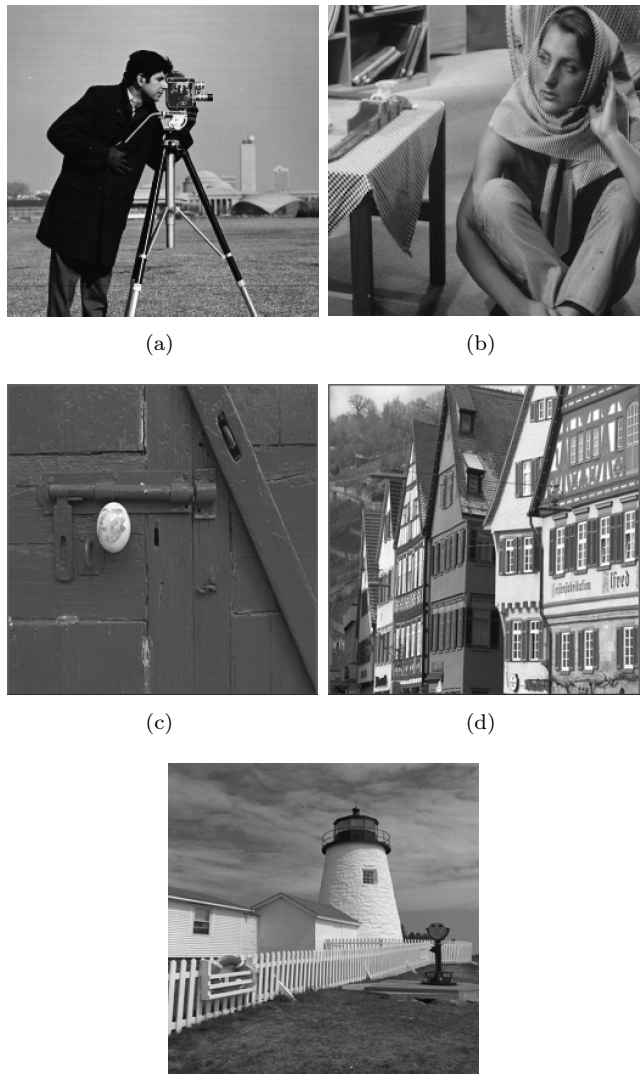


Figure 8: Original images for restoration simulation with the isotropic TV regularization, where (a) is Cameraman, (b) is Barbara, (c) is Kodim02, (d) is Kodim08, and (e) is Kodim19.

level. It is important to note that the MSE of each algorithm is close to the MSE without approximation (reference). CORDIC performs as well as the non-approximation for all patterns. However, the proposed method with

$N = 4$ also shows moderate performance with an average MSE about 10^{-6}. As an example, Figure 9 shows the results in terms of the MSE from the original image of Figure 8(a), where the standard deviation is $\sigma = 10/255$. In Figure 9, (a) and (b) show the original and observation image, respectively, (c) is a restored image with the original ℓ_2-norm evaluation, and (d) represents the result with the proposed approximation. Figure 9(d), (e), (f) are, respectively, the restored image for $N = 2$, 3, and 4 in (16). Comparing Figure 9(c) and (d), the difference in MSE is about 4.3×10^{-6}, indicating a good approximation by the proposed method.

As shown in Figure 9, the approximation by the proposed method performs higher than the original method. In some cases, our approximation shows a higher restoration performance than the original. Let's discuss the reasons. First, the cost function of the problem setting does not necessarily much to the quality assessment function, there is no guarantee of consistency between the optimality of the two functions. Second, the regularization term (priors) of the problem setting is not always appropriate for given images. The original regularization is isotropic, while our proposed approximation process slightly deviates from the isotropic. It cannot be denied that this anisotropy works appropriately as a prior for certain types of images, and it is possible that the restoration performance might improve contrary to our expectations. Figure 10 illustrates an example of image restoration by TV regularization in which this flipping phenomenon occurs. In this demonstration, we rotated an artificial stripe pattern image from 0 to $\pi/2$ radian and performed image restoration. This phenomenon suggests that deviating slightly from the isotropic process may be advantageous for anisotropic images.

4.4 FPGA implementation

Table 6 summarizes the FPGA implementation specifications in this study, where Xilinx's tools are used. The target board is Alveo U250 Data Center accelerator card that is designed to accelerate compute-intensive applications such as machine learning, data analytics, and video processing.[2] We use only the programmable logic (PL) unit of FPGA in this evaluation. The complie strategy to synthesize the logic circuit from Vitis Model Composer was partially changed, where the amount of BRAM used in FPGA, and the DSP utilization were changed to 0. The target FPGA is equipped with DSPs and BRAMs that can accelerate operations, however, DSPs and BRAMs that can only be used for specific tasks, there is no guarantee of availability when coexisting with other processes, and also it is not expected to be available in other HW

[2]Alveo U200 and U250 Data Center Accelerator Cards Data Sheet can be viewed at https://www.avnet.com/opasdata/d120001/medias/docus/190/XLX-A-U200-P64G-PQ-G-Datasheet.pdf (Oct 2018).

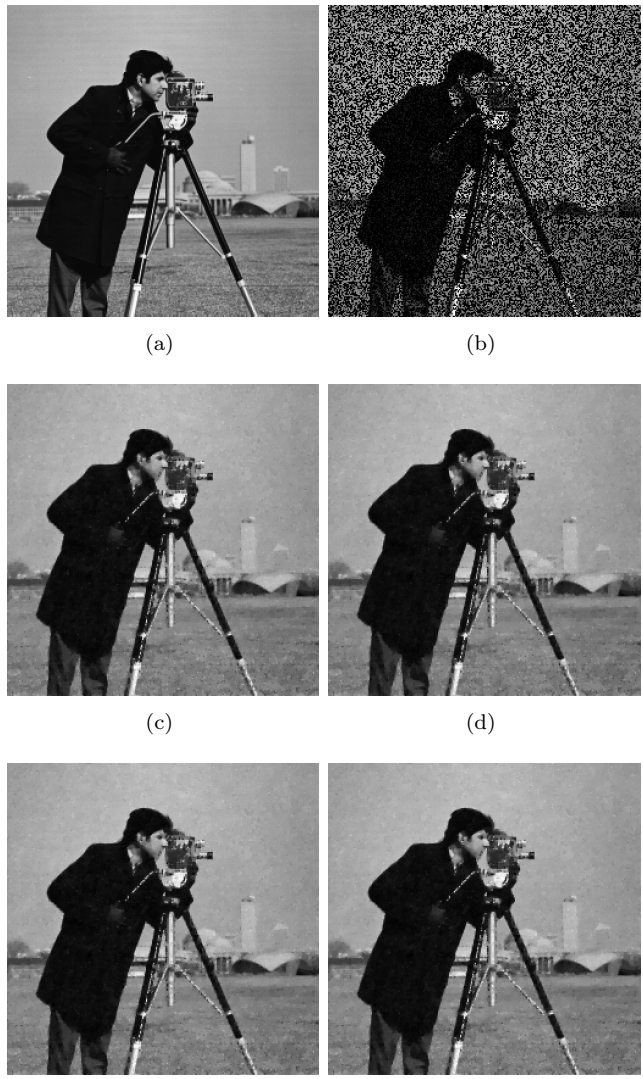
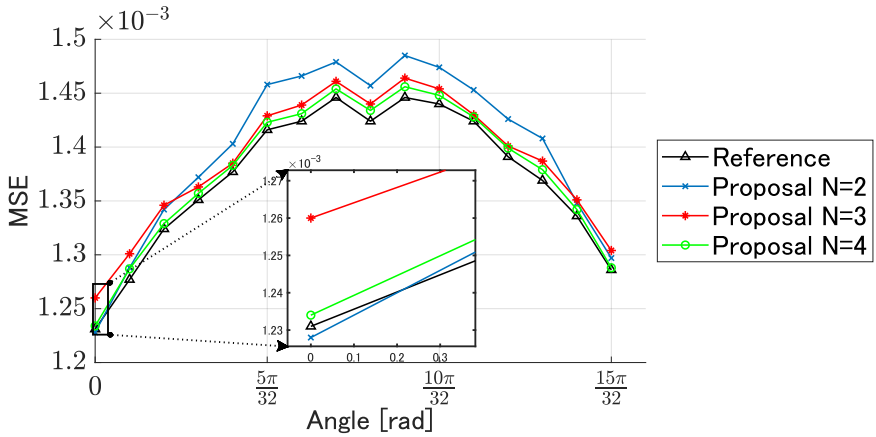
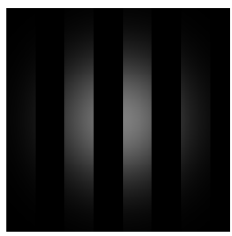


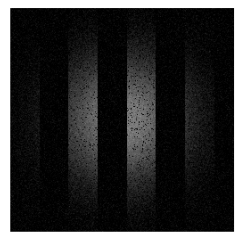
Figure 9: Simulation results of image restoration by means of isotropic TV regularization. (a) Original image, (b) Observation image (MSE: 0.132132), (c) Restored image without approximation (MSE: 0.002175), (d), (e), (f) Restored image with the proposed approximation for $N = 2$ (MSE: 0.002183), $N = 3$ (MSE: 0.002174) and, $N = 4$ (MSE: 0.002175) respectively.



(a)



(b)



(c)

Figure 10: Simulation results of image restoration for an artificial stripe pattern image rotated from 0 to $\pi/2$ radian, where the image size is 256×256, the standard deviation of AWGN is $10/255$ and the pixel loss rate is 5. (a) Comparison of MSEs, (b) Original image, and (c) Observation image.

implementations. Therefore, we set Vitis Model Composer block parameter implementation option not to use these resources in our experiments.

PCA and CORDIC are implemented as existing methods for references. The size of the truth table per input of PCA is shown as the number of nodes N_p. Since this implementation has 2-tuple inputs, the size of the truth table is the square of the number of nodes. In this implementation of the 2-tuple ℓ_2-norm operation, x_1 and x_2 can be swapped as $\sqrt{x_1^2 + x_2^2} = \sqrt{x_2^2 + x_1^2}$. Therefore, the scale of the final implemented truth table can be reduced to about half of the square of the number of nodes. The CORDIC IP has the

Table 6: FPGA implementation specifications.

FPGA Implementation Environment	Xilinx Vivado 2022.1
Target Board	Alveo U250 Data Center Accelerator Card
Target card	Virtex UltraScale+ FPGA
Synthesis strategy	Vivado Synthesis Default
Implementation strategy	Vivado Implementaion Default
HDL	Verilog
HDL code generation	Vitis Model Composer 2022.1
Vitis Model Composer Library block	HDL library block
Vitis Model Composer implementation Option	no DSP and BRAM

following implementation options: No Pipelining, Optimal, and Maximum. CORDIC options are as follows: [3]

- In No Pipelining, CORDIC core is implemented without pipelining.
- In Optimal, CORDIC core is implemented with as many stages of pipelining as possible without using any additional LUTs.
- In Maximum, CORDIC core is implemented with a pipeline after every shift-add sub stage.

Using Vitis Model Composer, we can automatically generate synthesizable HDL code mapped to the Xilinx optimized algorithms.[4]

Let us compare the accuracy in fixed-point arithmetic among the proposed method and the existing approximation methods (CORDIC and PCA). The proposed and existing methods were applied to (10), a 2-tuple signal in the range $[-0.9, 0.9]$, and the output results were compared with the computation results of the square root function of MATLAB in double precision data. The evaluation was performed in terms of MSE. The relation between the bit width and MSE for each method is shown in Figure 11. PCA with $N_p = 10$ has a small number of nodes and, its accuracy is lower than that of the other methods. The MSE of PCA with $N_p = 10$ is about 0.018, which is unacceptable for 8-bit image processing. Since the MSE of the methods other than PCA with $N_p = 10$ are not evaluated in Figure 11, Figure 12 shows the MSE for methods other than PCA with $N_p = 10$.

For the proposed method, the MSE decreases as the bit width increases from 8 to 12 bits, and for CORDIC the MSE decreases from 11 to 16 bits. No significant change was observed for any of the methods beyond 16 bits. For CORDIC, the MSE is close to zero due to the high arithmetic precision. The

[3]CORDIC IP Product Guide can be viewed at https://docs.amd.com/v/u/en-US/pg105-cordic (Aug 2021).

[4]Vitis Model Composer User Guide can be viewed at https://docs.amd.com/r/en-US/ug1483-model-composer-sys-gen-user-guide (Nov 2024).

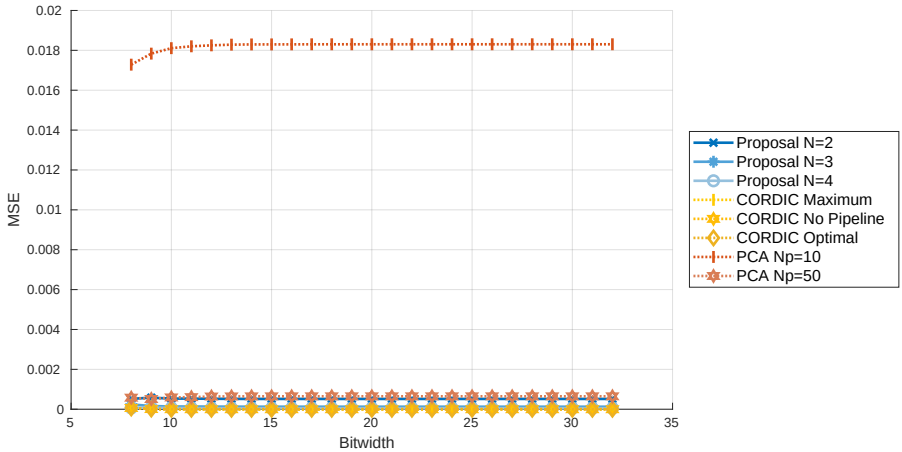


Figure 11: Simulation results of the computational accuracy of the proposed method and the fixed-point implementation of the existing method evaluated in terms of MSE, where N denotes the composite number of ℓ_1-cones of the proposed method, N_p of PCA is the number of nodes.

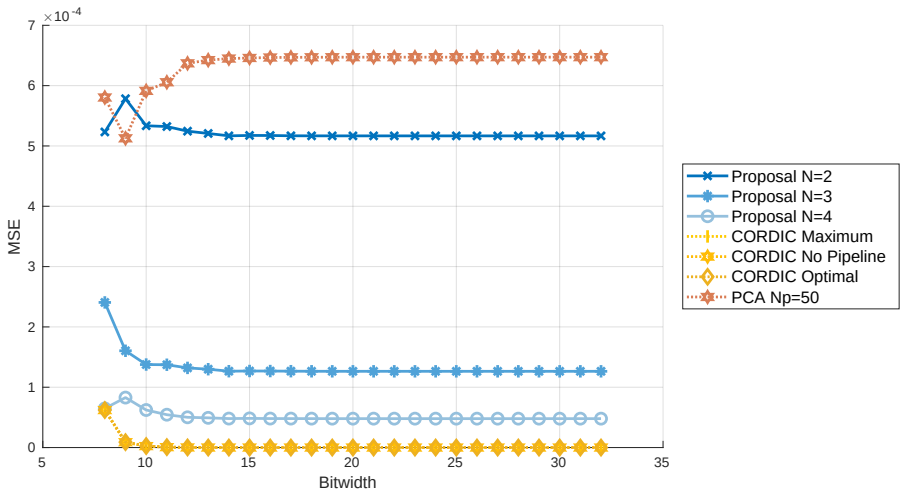


Figure 12: Enlarged view of part of Figure 11, where simulation results of the computational accuracy of the proposed method and the fixed-point implementation of the existing method other than PCA with $N_p = 10$ are shown in terms of MSE. N denotes the composite number of ℓ_1-cones of the proposed method, N_p denotes the number of nodes of PCA.

proposed method with $N = 3$ or 4 approaches to the CORDIC accuracy. The error between CORDIC and the proposed method with $N = 3$ is less than 2.0×10^{-4}, which we consider sufficient for 8-bit image processing.

4.5 Synthesis Report for FPGA

Let us report the synthesis result of (10) with the proposed method and existing methods for FPGA, and measure the performance.

Since the proposed method and PCA can approximate (10) directly, each method was implemented with a dedicated stand-alone circuit. In contrast, CORDIC is a method that can calculate square roots, but cannot calculate square. Therefore, we evaluate CORDIC combined with a circuit for the square operation.

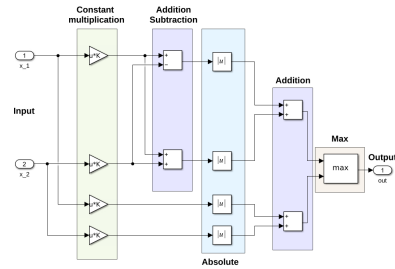
The amount of resources used in the FPGA and the timing report were obtained from the output reports produced after the synthesis by Vivado. Let us use vendor-supplied IPs for CORDIC as an existing method, and allow for a fair performance comparison with the proposed method. For the proposed method and the existing PCA have no existing IPs and, thus we use the Vitis Model Composer, which allows RTL design within MATLAB/Simulink environment. The output HDL code is checked and compared among the RTL model of the above three methods, CORDIC, PCA and the proposed method to evaluate the performance.

4.5.1 Composition of the Proposed Method

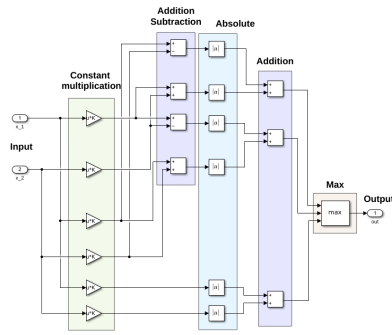
The components of the proposed method are described in detail. The design model consists of a circuit based on Algorithm 1 and implements on FPGA. A schematic block diagram of the proposed algorithm designed from Simulink is shown in Figure 13. First, we discuss the parameter values used within the proposed method, i.e., c_k and s_k in Algorithm 1, which can be computed in advance. The parameters for $N = 2, 3$, and 4 are as follows:

- For $N = 2$,
  $c_0 = 1/\sqrt{2}, c_1 = (1/\sqrt{2})\cos\pi/4, s_0 = 0, s_1 = (1/\sqrt{2})\sin\pi/4.$
- For $N = 3$,
  $c_0 = 1/\sqrt{2}, c_1 = (1/\sqrt{2})\cos\pi/6, c_2 = (1/\sqrt{2})\cos 2\pi/6,$
  $s_0 = 0, s_1 = (1/\sqrt{2})\sin\pi/6, s_2 = (1/\sqrt{2})\sin 2\pi/6.$
- For $N = 4$,
  $c_0 = 1/\sqrt{2}, c_1 = (1/\sqrt{2})\cos\pi/8, c_2 = (1/\sqrt{2})\cos 2\pi/8,$
  $c_3 = (1/\sqrt{2})\cos 3\pi/8, s_0 = 0, s_1 = (1/\sqrt{2})\sin\pi/8,$
  $s_2 = (1/\sqrt{2})\sin 2\pi/8, s_3 = (1/\sqrt{2})\sin 3\pi/8.$

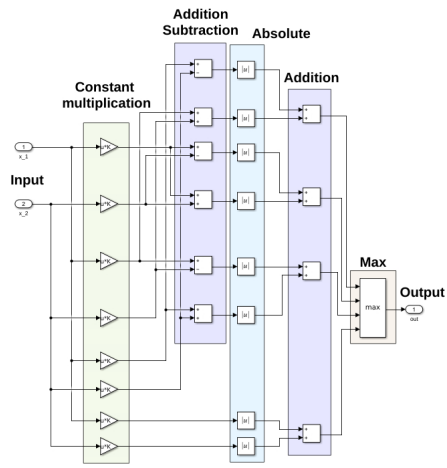
These constants are calculated in advance, and a_k in Algorithm 1 is obtained only by constant multiplication (scaling), addition, subtraction, and absolute operation. The proposed method consists only of constant multiplication, addition/subtraction, absolute operation, and MAX calculations, no matter how much the composite number of ℓ_1-cones N is increased. All operations can be computed in one clock cycle. The number of operations is $2N$ for



(a)



(b)



(c)

Figure 13: A schematic block diagram of the proposed algorithm. (a), (b), and (c) are the proposed approximation for $N = 2, N = 3$, and $N = 4$, respectively.

constant multiplications, $3N-2$ for additions or subtractions, $2N$ for absolute operation, and one MAX operation for N values.

Next, let us explain the bit width of each stage for the 16 bit inputs. In the followings, the bit widths are represented in the Q-format, where

$\mathbf{Q}_{m,n}$ Format with the m-bit signed integer part and n-bit fractional part.

$\mathbf{UQ}_{m,n}$ Format with the m-bit unsigned integer part and n-bit fractional part.

The design model with FPGA implementation from Vitis Model Composer is shown in Figure 14. In Figure 14, boxes in the same color share the same bit widths for all $N \in \{2,3,4\}$. The constant multiplication, addition/subtraction, absolute operation, and multiplexing were computed with the Simulink Full precision. The bit width from the input to the output is as follows:

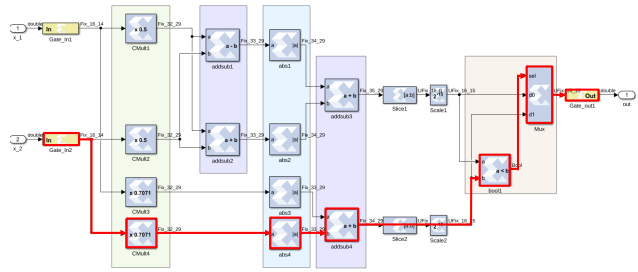
- The input values x_1 and x_2 are assumed to be $x_i \in [-1,1]$: Q1.14
- The first constant multiplication (green box): Q2.29
- The addition/subtraction (purple box): Q3.29
- The absolute value (blue box): Q4.29
- The addition (purple box): Q5.29
- In the process of computing c_0, s_0, the absolute value (blue box): Q3.29, and the addition (purple box): Q4.29
- The slice block to cut a sequence of bits from the input data and the scale block to scale the input by a power of 2: UQ1.15
- The multiplexer and the comparison function: UQ1.15

In this way, the proposed algorithm performs norm computation for a 16-bit input.

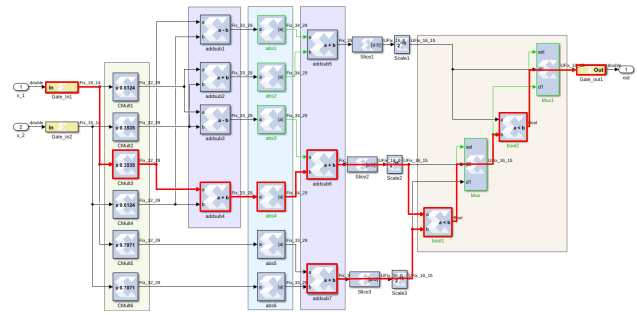
4.5.2 Latency

This paper defines latency as "the time from the first input to the first output." Low latency is preferable, meaning little delay time.

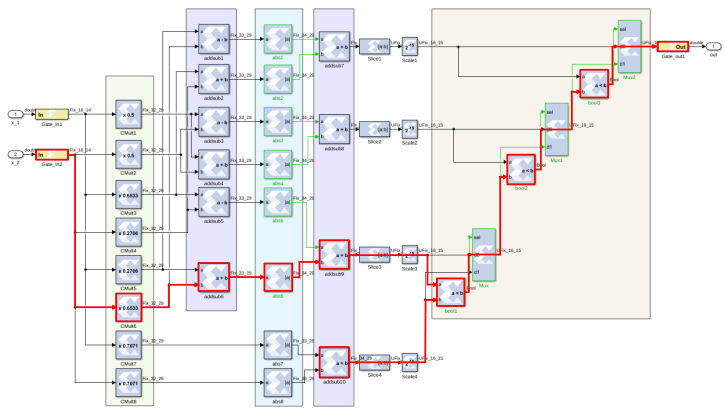
First, we obtain the number of clocks taken from the first input to the first output from the Integrated Logic Analyzer (ILA) waveform observation results. The proposed method and the existing methods; CORDIC and PCA, are implemented with 16 bit width. Figure 15 illustrates the waveform. The ILA waveforms were obtained based on the architecture shown in Figure 14. The waveform at the top of the Figure 15 shows the clock signal with period 50 ns. The proposed method, PCA and CORDIC (option: No Pipelining) takes 1 clock to output. CORDIC (option: Optimal) takes a total of 9 clocks



(a)



(b)



(c)

Figure 14: The design model with FPGA implementation from Vitis Model Composer. (a), (b), and (c) are the proposed approximation for $N = 2, N = 3$, and $N = 4$, respectively. The red line indicates the critical path.

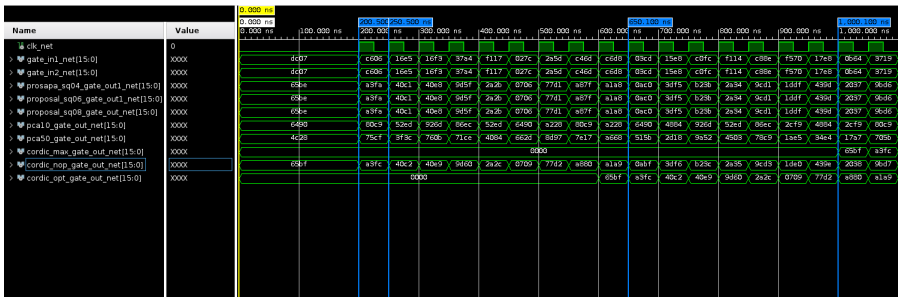


Figure 15: Waveform observation results by Integrated Logic Analyzer (ILA) when input signal is supplied to FPGA. The proposed method, PCA and CORDIC are modeled for FPGA with 16 bit width. The clock period was set to 50 ns. The 1st line is the clock, the 2nd and 3rd lines are the input signals x_1 and x_2, the lines from the 4th to 6th are the waveforms of the proposed method ($N = 2$, 3, and 4), the 7th and 8th lines are the waveforms of the PCA ($N_p = 10$, 50), and the lines from the 9th to 11th are the waveforms of CORDIC (option: Maximum, No Pipelining, Optimal).

to output. CORDIC (option: Maximum) takes a total of 17 clocks to output. Therefore, the proposed method can output the results with fewer or equivalent number of clocks compared with the other methods.

Next, we evaluate the maximum operating frequency of the proposed and existing methods using $F_{\max}$. $F_{\max}$ is calculated as

$$F_{\max} = \frac{1}{T - T_W}, \tag{18}$$

where T is the FPGA clock period and T_W is the worst negative slack [39]. We implemented each method separately on FPGA with fixed $T = 50$ ns. The result of T_W is obtained from the synthesis report, and calculated $F_{\max}$ using (18). The higher $F_{\max}$ is, the higher frequency the method can run on FPGA. The critical paths are labeled by red lines in Figure 14. The critical paths were drawn based on the FPGA timing report available after the implementation of Vivado. The $F_{\max}$ values of the proposed method and existing methods are shown in Figure 16 for the 16-bit width case. N_p in PCA is the number of nodes. Figure 16 shows that CORDIC with Optimal or Maximum has higher $F_{\max}$ than the other methods, followed by the proposed method, CORDIC with No Pipelining and PCA.

Finally, we calculated the latency from the ILA waveform in Figure 15 and the $F_{\max}$ in Figure 16 using the following equation:

$$T_{\text{latency}} = \frac{P_{\text{clks}}}{F_{\max}}, \tag{19}$$

where P_{clks} is the number of clocks taken from the first input to the first output. Figure 17 summarizes the latencies, where we see that the proposed

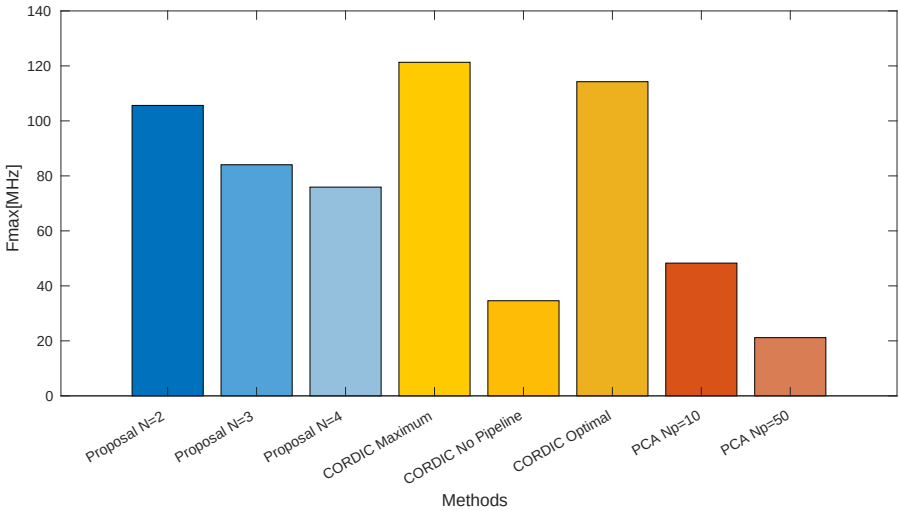


Figure 16: The bars show $F_{\max}$, which is the maximum operable frequency of FPGA. The proposed method and existing method are implemented with 16-bit width. N_p in the PCA method indicates the number of nodes. No Pipelining, Optimal, and Maximum in CORDIC label indicate the implementation options of Xilinx LogiCORE CORDIC IP.

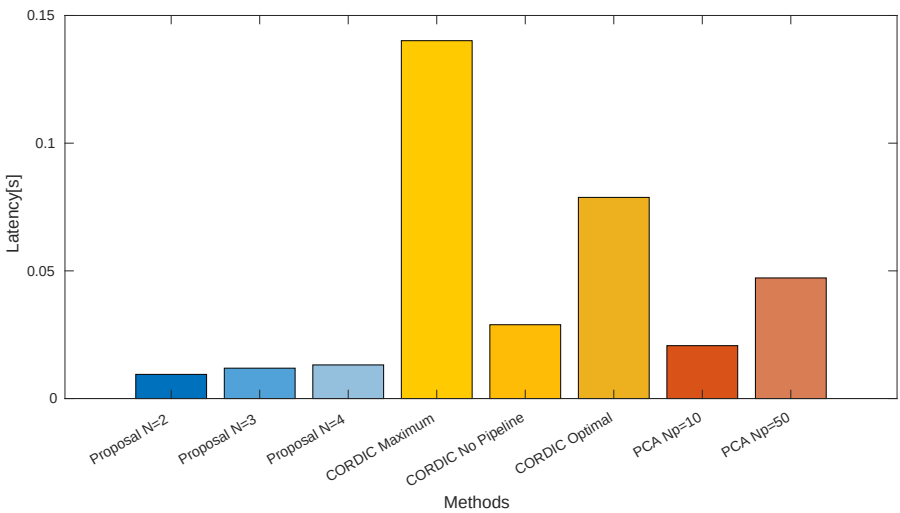


Figure 17: The bars show latency, which is the time taken from the first input to the first output. The proposed and existing methods are implemented with 16-bit width.

method has smaller latency than the existing methods for all N, followed by CORDIC and PCA.

4.5.3 Throughput

This paper defines a throughput as "the reciprocal of the maximum processing time to consume from one output to the next one." A high throughput is preferable, meaning that processing speed is high.

First, we evaluate the number of the clocks from one output to the next one. Figure 18 shows the resulting waveforms, which are examined. All of the proposed and existing methods take one clock for the successding outputs. CORDIC usually operates a coarse rotation. However, the square root function does not require any coarse rotation. This is because, in the square root configuration, the CORDIC algorithm performs operations only in the first quadrant.

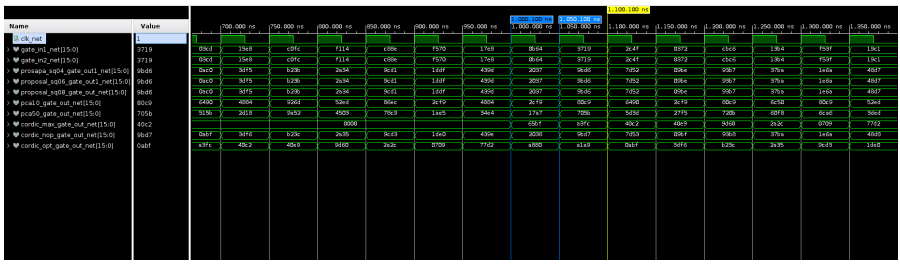


Figure 18: Waveform results by Integrated Logic Analyzer (ILA). The proposed method, PCA and CORDIC are modeled for FPGA with 16 bit width. The clock period was set to 50 ns. The 1st line is the clock, the 2nd and 3rd lines are the input signals x_1 and x_2, the lines from the 4th to 6th are the waveforms of the proposed method ($N = 2$, 3, and 4), the 7th and 8th lines are the waveforms of the PCA ($N_p = 10$, 50), the lines from the 9th to 11th are the waveforms of CORDIC (option: Maximum, No Pipelining, Optimal).

Next, the throughput is calculated from the maximum operable frequency $F_{\max}$ shown in Figure 16 with (18) by

$$T_{\text{throughput}} = \frac{F_{\max}}{R_{\text{clks}}}, \tag{20}$$

where R_{clks} is the number of clocks between successive outputs. The throughput results are shown in Figure 19. Figure 19 shows the reciprocal of the throughput defined in (20). CORDIC Maximum and CORDIC Optimal take the least amount of time, followed by the proposed method for all N, CORDIC No Pipelining, and PCA.

4.5.4 Circuit Area

Let us compare the circuit area among the proposed and existing methods. We summarize the arithmetic part of (10) for each method in Figure 20(a)

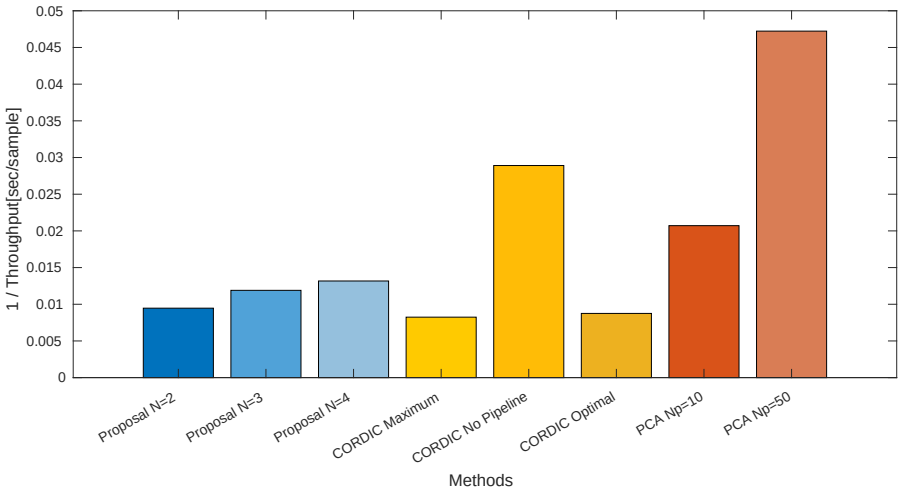
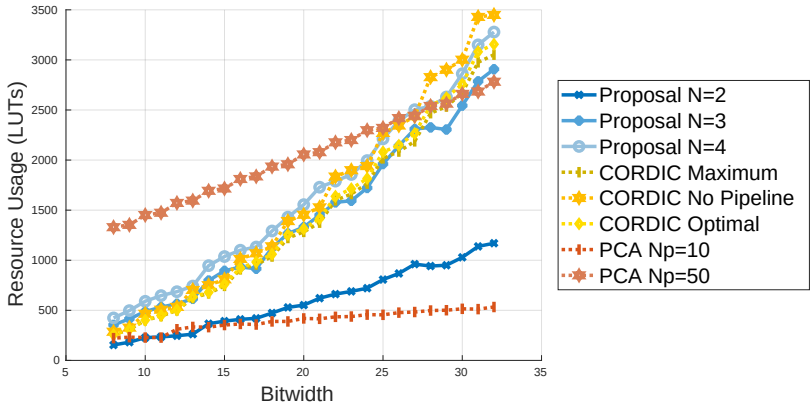


Figure 19: The bars show the evaluated maximum time from one output to the next one, i.e., the reciprocal of throughputs, where the throughput is defined by (20). The proposed and existing methods are implemented with 16-bit width.

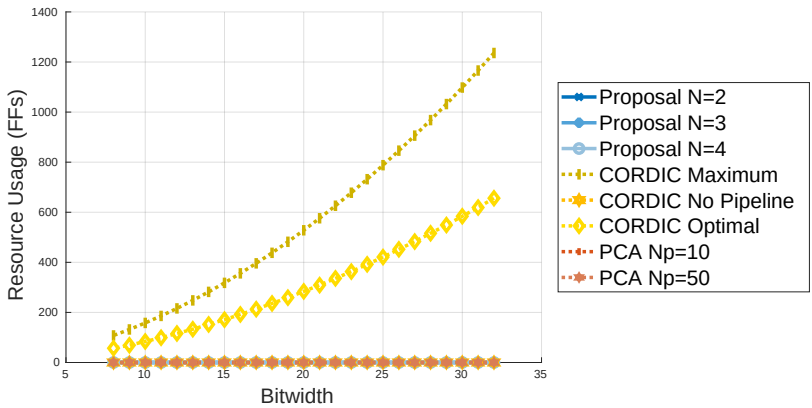
and 20(b). A look up table (LUT) is a truth table in FPGA. The lesser the amount of LUTs, the lesser the power consumption of the circuit can work. PCA with $N_p = 10$ has the lowest LUTs utilization, however its low arithmetic accuracy makes itself impractical as discussed in Section 4.4. Comparing with Figure 20(a), when $N = 2$ and $N = 3$, the proposed method requests less circuit area than the other practical existing methods. For $N = 4$, the proposed method requires a comparable amount of circuit resources as the existing CORDIC and PCA with $N_p = 50$. A FF is a storage element for sequential circuits in FPGA. Comparing with Figure 20(b), the proposed method and PCA uses no FF for any composite number N or N_p. The amount of FFs used in CORDIC is large because CORDIC requires repetitive computation and uses FFs as storage resource.

4.6 Comprehensive Evaluation

Since the relationship among calculation accuracy and circuit area, latency, and throughput has a trade-off, we visualize the performances and comprehensively evaluate the characteristics of each method to clarify the contribution of our proposed method. Table 7 summarizes their performances at 16-bit fixed-point. The resuls are collected from Sections 4.4, 4.5.2, 4.5.3, and 4.5.4. Values in Table 7 are ideal if they are close to zero. CORDIC has high arithmetic accuracy for any of the three methods. CORDIC Maximum and CORDIC Optimal implement pipelined processing with FF to achieve the



(a)



(b)

Figure 20: Synthesis reports with respect to the circuit area, where (a) shows the usage of look-up tables (LUTs) and (b) plots that of flip-flops (FFs) for the proposed and existing methods on FPGA, respectively. N indicates the composite number of ℓ_1-cones of the proposed method. N_p indicates the number of nodes in the PCA method. No Pipelining, Optimal, and Maximum in CORDIC label indicate the implementation options of Xilinx LogiCORE CORDIC IP. Proposal's FFs usage is zero for all bit widths.

lowest throughput compared to the other methods. CORDIC No Pipelining consists of only LUTs and offers low latency and moderate throughput. PCA with $N_p = 10$ has a small circuit area, moderate latency, and moderate throughput, however it shows a large MSE and low arithmetic accuracy. PCA with $N_p = 50$ has a higher accuracy than that with $N_p = 10$, while the LUT, latency, and throughput are worse significantly. Although increasing the number of nodes N_p to 50 or more is expected to improve the calculation

Table 7: Comparison among the proposed and existing methods, where all methods are configured with the 16 fixed-point bits. Each evaluation is taken from the results of the previous sections.

	MSE	Latency	1/Throughput	LUT	FF
Proposal $N = 2$	5.17×10^{-4}	**0.0095**	0.0095	410	**0**
Proposal $N = 3$	1.27×10^{-4}	0.0119	0.0119	933	**0**
Proposal $N = 4$	4.81×10^{-5}	0.0132	0.0132	1100	**0**
CORDIC Maximum	6.89×10^{-10}	0.1401	**0.0082**	909	356
CORDIC No Pipelining	$\mathbf{6.87 \times 10^{-10}}$	0.0289	0.0289	1013	**0**
CORDIC Optimal	6.86×10^{-10}	0.0788	0.0088	923	192
PCA $N_p = 10$	1.83×10^{-2}	0.0207	0.0207	**364**	**0**
PCA $N_p = 50$	6.47×10^{-4}	0.0472	0.0472	1811	**0**

accuracy, other performance indices are not favorable. In contract, the proposed method has moderate arithmetic accuracy with less circuit resources. Since the implementation does not use pipelining, it achieves low latency with moderate throughput. From the above discussion, CORDIC is appropriate when high calculation accuracy is required, however, while the proposed method can achieve high processing speed with low resource usage when a certain degree of error is tolerated.

5 Conclusions

In this paper, we proposed an ℓ_2-norm evaluation method for 2-tuple arrays, which is suitable for fixed-point arithmetic. We assessed the application by applying it to image restoration and evaluated the HW resource utilization and processing speed through the FPGA design. The degradation of the proposed approximation is verified to be insignificant with lower resource utilization and faster response time than other conventional techniques. The 2-tuple ℓ_2-norm evaluation method is used not only for TV regularization but also for complex ℓ_1-norm. Therefore, it has a wide range of applications. Our proposed algorithm focuses on 2-tuple elements. On the other hand, we are aware that there are important applications for higher tuples, such as 3-dimensional total variation. For example, it is possible to apply our method to the computation of the high-tuple ℓ_2-norm by implementing the approximation in (16) in a hierarchical manner, but there is no guarantee that this approach is efficient, and a new architecture needs to be proposed, analyzed, and evaluated. Since this is beyond the scope of this paper, we leave it as an open problem. In the future, we will apply the proposed 2-tuple ℓ_2-norm method to image restoration problems other than denoising and extend it to 3-tuples and more.

Appendix

A Derivation of Performance Estimation Functions

In the following, we summarize the derivation process of the equations in Table 2.

A.1 Proposed Method

Let us derive $\mathcal{A}_{\text{pro}}^{(\text{b})}$, $\mathcal{R}_{\text{pro}}^{(\text{b})}$, $\mathcal{L}_{\text{pro}}^{(\text{b})}$, and $\mathcal{T}_{\text{pro}}^{(\text{b})}$ in Table 2.

Derivation of $\mathcal{A}_{\text{pro}}^{(\text{b})}$

The arithmetic operations required for the norm calculation are listed as follows, where Figure 13 is used as a reference.

- $\alpha_{\text{add}}^{(\text{b})}$: $3N-2$ modules
  Three adders are required to obtain a_k in Algorithm 1. When $k=0$, only one addition is needed, so $3N-2$ adders (addition modules) are required.
- $\alpha_{\text{scaling}}^{(\text{b})}$: $2N$ modules
  Looking at Algorithm 1, it appears that constant multiplier is required $4N$ modules, but the circuit can be reused when the results of sin and cos calculations match. Therefore, $2N$ multipliers (multiplication modules) are required.

As a result, we have $\mathcal{A}_{\text{pro}}^{(\text{b})} = (3N-2)\alpha_{\text{add}}^{(\text{b})} + 2N\alpha_{\text{scaling}}^{(\text{b})}$.

Derivation of $\mathcal{R}_{\text{pro}}^{(\text{b})}$

The proposed method does not use any register during the norm calculation. We only assume one register after the norm calculation. As a result, we have $\mathcal{R}_{\text{pro}}^{(\text{b})} = \rho_{\text{ff}}^{(\text{b})}$.

Derivation of $\mathcal{L}_{\text{pro}}^{(\text{b})}$

The arithmetic operations required to reach to the output are shown below, where Figure 13 is used as a reference.

- $\xi_{\text{add}}^{(\text{b})}$: 2 modules
- $\xi_{\text{scaling}}^{(\text{b})}$: 1 module

From Algorithm 1 and Figure 13, the input requires one multiplication and two additions to reach to the output. Assuming one register after the norm computation, the setup and hold time is required before output. As a result, we have $\mathcal{L}_{\text{pro}}^{(\text{b})} = 2\xi_{\text{add}}^{(\text{b})} + \xi_{\text{scaling}}^{(\text{b})} + \delta_{\text{ff}}$.

Derivation of $\mathcal{T}_{\text{pro}}^{(\text{b})}$

The arithmetic operations required to reach to the output are shown below, where Figure 13 is used as a reference.

- $\xi_{\text{add}}^{(\text{b})}$: 2 modules
- $\xi_{\text{scaling}}^{(\text{b})}$: 1 module

From Algorithm 1 and Figure 13, the input requires one multiplication and two additions to reach to the output. Assuming one register after the norm computation, the setup and hold time is required before output. As a result, we have $\mathcal{T}_{\text{pro}}^{(\text{b})} = 2\xi_{\text{add}}^{(\text{b})} + \xi_{\text{scaling}}^{(\text{b})} + \delta_{\text{ff}}$.

A.2 CORDIC without Pipeline

Let us derive $\mathcal{A}_{\text{nop}}^{(\text{b})}$, $\mathcal{R}_{\text{nop}}^{(\text{b})}$, $\mathcal{L}_{\text{nop}}^{(\text{b})}$, and $\mathcal{T}_{\text{nop}}^{(\text{b})}$ in Table 2.

Derivation of $\mathcal{A}_{\text{nop}}^{(\text{b})}$

CORDIC computes the norm by combining with the square operation. The arithmetic operations required for the norm calculation are as follows.

- $\alpha_{\text{add}}^{(\text{b})}$: 1 module
- $\alpha_{\text{mult}}^{(\text{b})}$: 2 modules
  Functions required for performing a square operation, $x_{i,1}^2 + x_{i,2}^2$.
- $\alpha_{\text{add}}^{(\text{b})}$: 1 module
  CORDIC internal operation to calculate the square root. CORDIC requires addition and shifting of input bits. Repeated use of one adder.

As a result, we have $\mathcal{A}_{\text{nop}}^{(\text{b})} = 2\alpha_{\text{add}}^{(\text{b})} + 2\alpha_{\text{mult}}^{(\text{b})}$.

Derivation of $\mathcal{R}_{\text{nop}}^{(\text{b})}$

CORDIC without Pipeline does not use any register during the norm calculation. We assume only one register after the norm calculation. As a result, we have $\mathcal{R}_{\text{nop}}^{(\text{b})} = \rho_{\text{ff}}^{(\text{b})}$.

Derivation of $\mathcal{L}_{\mathrm{nop}}^{(\mathrm{b})}$

The path required to reach to the output are shown below.

- $\xi_{\mathrm{add}}^{(\mathrm{b})}$: 1 module
- $\xi_{\mathrm{mult}}^{(\mathrm{b})}$: 1 module
  The path of the process of computing a square operation.
- $\xi_{\mathrm{add}}^{(\mathrm{b})}$: b modules
  It must be passed through adders with an input bit width of b inside CORDIC before reaching output.

Assuming one register after the norm computation, the setup and hold time is required before output. As a result, we have $\mathcal{L}_{\mathrm{nop}}^{(\mathrm{b})} = (b+1)\xi_{\mathrm{add}}^{(\mathrm{b})} + \xi_{\mathrm{mult}}^{(\mathrm{b})} + \delta_{\mathrm{ff}}$.

Derivation of $\mathcal{T}_{\mathrm{nop}}^{(\mathrm{b})}$

The path required to reach to the output are shown below.

- $\xi_{\mathrm{add}}^{(\mathrm{b})}$: 1 module
- $\xi_{\mathrm{mult}}^{(\mathrm{b})}$: 1 module
  The path of the process of computing a square operation.
- $\xi_{\mathrm{add}}^{(\mathrm{b})}$: b modules
  It must be passed through adders with an input bit width of b inside CORDIC before reaching output.

Assuming one register after the norm computation, the setup and hold time is required before output. As a result, we have $\mathcal{T}_{\mathrm{nop}}^{(\mathrm{b})} = (b+1)\xi_{\mathrm{add}}^{(\mathrm{b})} + \xi_{\mathrm{mult}}^{(\mathrm{b})} + \delta_{\mathrm{ff}}$.

A.3 CORDIC with Pipeline

Let us derive $\mathcal{A}_{\mathrm{pip}}^{(\mathrm{b})}$, $\mathcal{R}_{\mathrm{pip}}^{(\mathrm{b})}$, $\mathcal{L}_{\mathrm{pip}}^{(\mathrm{b})}$, and $\mathcal{T}_{\mathrm{pip}}^{(\mathrm{b})}$ in Table 2.

Derivation of $\mathcal{A}_{\mathrm{pip}}^{(\mathrm{b})}$

CORDIC computes the norm by combining with the square operation. The arithmetic operations required for the norm calculation are as follows.

- $\alpha_{\mathrm{add}}^{(\mathrm{b})}$: 1 module
- $\alpha_{\mathrm{mult}}^{(\mathrm{b})}$: 2 modules
  Functions required for performing a square operation, $x_{i,1}^2 + x_{i,2}^2$.

- $\alpha_{\mathrm{add}}^{(\mathrm{b})}$: b modules
  Adders are lined up b modules on the pipeline.

As a result, we have $\mathcal{A}_{\mathrm{pip}}^{(\mathrm{b})} = (b+1)\alpha_{\mathrm{add}}^{(\mathrm{b})} + 2\alpha_{\mathrm{mult}}^{(\mathrm{b})}$.

Derivation of $\mathcal{R}_{\mathrm{pip}}^{(\mathrm{b})}$

CORDIC with pipeline requires b internal pipeline registers of input bit widths b. The number of the registers should be the number of pipeline stages. If all the adders are pipelined, the number will be b. As a result, we have $\mathcal{R}_{\mathrm{pip}}^{(\mathrm{b})} = b\rho_{\mathrm{ff}}^{(\mathrm{b})}$.

Derivation of $\mathcal{L}_{\mathrm{pip}}^{(\mathrm{b})}$

The path required to reach to the output are shown below.

- $\xi_{\mathrm{add}}^{(\mathrm{b})}$: 1 module
- $\xi_{\mathrm{mult}}^{(\mathrm{b})}$: 1 module
  The path of the process of computing a square operation.
- $\xi_{\mathrm{add}}^{(\mathrm{b})}$: b modules
  Adders are lined up b modules on the pipeline.

Assuming that the pipeline has a number of bit- width stages, b setup and hold times are required before output. As a result, we have $\mathcal{L}_{\mathrm{pip}}^{(\mathrm{b})} = (b+1)\xi_{\mathrm{add}}^{(\mathrm{b})} + \xi_{\mathrm{mult}}^{(\mathrm{b})} + b\delta_{\mathrm{ff}}$.

Derivation of $\mathcal{T}_{\mathrm{pip}}^{(\mathrm{b})}$

The path required to reach to the output are shown below.

- $\xi_{\mathrm{add}}^{(\mathrm{b})}$: 1 module
- $\xi_{\mathrm{mult}}^{(\mathrm{b})}$: 1 module
  The path of the process of computing a square operation.

Consider the setup and hold times of the pipeline process. As a result, we have $\mathcal{T}_{\mathrm{pip}}^{(\mathrm{b})} = \xi_{\mathrm{add}}^{(\mathrm{b})} + \xi_{\mathrm{mult}}^{(\mathrm{b})} + \delta_{\mathrm{ff}}$.

A.4 PCA

Let us derive $\mathcal{A}_{\mathrm{pca}}^{(\mathrm{b})}$, $\mathcal{R}_{\mathrm{pca}}^{(\mathrm{b})}$, $\mathcal{L}_{\mathrm{pca}}^{(\mathrm{b})}$, and $\mathcal{T}_{\mathrm{pca}}^{(\mathrm{b})}$ in Table 2.

Derivation of $\mathcal{A}_{\mathrm{pca}}^{(\mathrm{b})}$

The arithmetic operations required for the norm calculation are as follows.

- $\alpha_{\mathrm{add}}^{(\mathrm{b})}$: $n+1$ modules
  PCA uses n additions to identify the breakpoint vectors for input values. It is estimated to be $n+1$, taking into account the fact that the cost increases in stages for the two inputs as $\frac{2}{n}(1+2+\cdots+n)\alpha_{\mathrm{add}}$.
- $\alpha_{\mathrm{add}}^{(\mathrm{b})}$: 3 modules
- $\alpha_{\mathrm{scaling}}^{(\mathrm{b})}$: 1 module
- $\alpha_{\mathrm{mult}}^{(\mathrm{b})}$: 1 module
  PCA requires three additions, one constant multiplication, and one variable multiplication to obtain table data.

Since PCA's input values are used only for comparison operations, addition and multiplication do not depend on the input bits as much as other methods. As a result, we have $\mathcal{A}_{\mathrm{pca}}^{(\mathrm{b})} = (n+4)\alpha_{\mathrm{add}}^{(\mathrm{b})} + \alpha_{\mathrm{scaling}}^{(\mathrm{b})} + \alpha_{\mathrm{mult}}^{(\mathrm{b})}$.

Derivation of $\mathcal{R}_{\mathrm{pca}}^{(\mathrm{b})}$

PCA does not use any register in the norm calculation. We assume only one register after the norm calculation. As a result, we have $\mathcal{R}_{\mathrm{pca}}^{(\mathrm{b})} = \rho_{\mathrm{ff}}^{(\mathrm{b})}$.

Derivation of $\mathcal{L}_{\mathrm{pca}}^{(\mathrm{b})}$

The path required to reach to the output are shown below.

- $\alpha_{\mathrm{add}}^{(\mathrm{b})}$: n modules
  PCA uses n additions to identify the breakpoint vectors for input values x.
- $\alpha_{\mathrm{add}}^{(\mathrm{b})}$: 3 modules
- $\alpha_{\mathrm{scaling}}^{(\mathrm{b})}$: 1 module
- $\alpha_{\mathrm{mult}}^{(\mathrm{b})}$: 1 module
  PCA requires three additions, one constant multiplication, and one variable multiplication to obtain table data.

Assuming one register after the norm computation, the setup and hold time is required before output. As a result, we have $\mathcal{L}_{\text{pca}}^{(\text{b})} = (n+3)\xi_{\text{add}}^{(\text{b})} + \xi_{\text{scaling}}^{(\text{b})} + \xi_{\text{mult}}^{(\text{b})} + \delta_{\text{ff}}$.

Derivation of $\mathcal{T}_{\text{pca}}^{(\text{b})}$

The path required to reach to the output are shown below.

- $\alpha_{\text{add}}^{(\text{b})}$: n modules
  PCA uses n additions to identify the breakpoint vectors for input values x.
- $\alpha_{\text{add}}^{(\text{b})}$: 3 modules
- $\alpha_{\text{scaling}}^{(\text{b})}$: 1 module
- $\alpha_{\text{mult}}^{(\text{b})}$: 1 module
  PCA requires three additions, one constant multiplication, and one variable multiplication to obtain table data.

Assuming one register after the norm computation, the setup and hold time is required before output. As a result, we have $\mathcal{T}_{\text{pca}}^{(\text{b})} = (n+3)\xi_{\text{add}}^{(\text{b})} + \xi_{\text{scaling}}^{(\text{b})} + \xi_{\text{mult}}^{(\text{b})} + \delta_{\text{ff}}$.

B PDS Algorithm for the TV Regularized Image Restoration

In the following, we describe the PDS algorithm in detail [9, 28]. First, transformation of (3) using the indicator function $\iota_C(\cdot)$ can be formulated as

$$\hat{\mathbf{u}} = \arg\min_{\mathbf{u}} \frac{1}{2}\|\mathbf{P}\mathbf{u} - \mathbf{v}\|_2^2 + \lambda\|\boldsymbol{\Delta}\mathbf{u}\|_{1,2} + \iota_{[u_{\min}, u_{\max}]^n}(\mathbf{u}). \tag{21}$$

The problem in (21) is applicable to PDS. PDS is an algorithm that solves an optimization problem in the form

$$\hat{\mathbf{u}} = \arg\min_{\mathbf{u}\in[u_{\min}, u_{\max}]^n} f(\mathbf{u}) + g(\mathbf{u}) + h(\mathbf{L}\mathbf{u}), \tag{22}$$

where $f\colon \mathbb{R}^n \to \mathbb{R}\cup\{\infty\}$, $g\colon \mathbb{R}^n \to \mathbb{R}\cup\{\infty\}$, and $h\colon \mathbb{R}^n \to \mathbb{R}\cup\{\infty\}$ are lower semicontinuous closed proper convex function, f is differentiable, and ∇f is β-Lipschitz continuous, where $\mathbf{L} \in \mathbb{R}^{l\times n}$. To apply the PDS algorithm to problem (21), we define

$$f(\mathbf{u}) = \frac{1}{2}\|\mathbf{P}\mathbf{u} - \mathbf{v}\|_2^2, \tag{23a}$$

$$g(\mathbf{u}) = \iota_{[u_{\min}, u_{\max}]^n}(\mathbf{u}), \tag{23b}$$

$$h(\mathbf{L}\mathbf{u}) = \lambda\|\boldsymbol{\Delta}\mathbf{u}\|_{1,2}, \tag{23c}$$

where $\mathbf{L} = \mathbf{\Delta}$. The PDS algorithm steps are derived as

$$\mathbf{u}^{(k+1)} = P_C\left(\mathbf{u}^{(k)} - \gamma_1\left(\lambda\mathbf{P}^\top(\mathbf{P}\mathbf{u}^{(k)} - \mathbf{v}) + \mathbf{\Delta}^\top\mathbf{z}^{(k)}\right)\right), \tag{24}$$

$$\widetilde{\mathbf{z}}^{(k+1)} = \mathbf{z}^{(k)} + \gamma_2\mathbf{\Delta}\left(2\mathbf{u}^{(k+1)} - \mathbf{u}^{(k)}\right), \tag{25}$$

$$\mathbf{z}^{(k+1)} = \widetilde{\mathbf{z}}^{(k+1)} - \mathcal{S}\left(\widetilde{\mathbf{z}}^{(k+1)}\right), \tag{26}$$

with some initial state, e.g., $k \leftarrow 0$, $\mathbf{u}^{(0)} = \mathbf{v}$ and $\mathbf{z}^{(0)} = \mathbf{\Delta}\mathbf{v}$, where $P_C(\cdot)$ denotes the metric projection onto $C = [u_{\min}, u_{\max}]^n$, $\mathbf{z}$ and $\widetilde{\mathbf{z}}$ denote intermediate variables, γ_1 and γ_2 are step sizes that should satisfy $\gamma_1^{-1} - \gamma_2(\sigma_{\max}(\mathbf{\Delta}))^2 \geq \frac{\beta}{2}$. $\sigma_{\max}(\mathbf{\Delta})$ denotes the maximum singular value of $\mathbf{\Delta}$. $\mathcal{S}(\cdot)$ in (26) represents soft-thresholding for magnitude of 2-tuples in $\mathbf{x}$ as in (5), i.e.,

$$[\mathcal{S}(\mathbf{x})]_{\mathcal{I}_i} = [\mathbf{x}]_{\mathcal{I}_i} \odot \max\left(\mathbf{1} - \frac{1}{\|[\mathbf{x}]_{\mathcal{I}_i}\|_2}\mathbf{1}, \mathbf{0}\right),$$

where $\mathcal{I}_i$ denotes the index set of the 2-tuple of the i-th pixel, i.e., the indices of the horizontal and vertical gradient components.

Biographies

Shu ABE received the B.S. degree in Electrical and Electronic Engineering from Niigata University, Japan in 2023. He is currently a M.E. candidate at Niigata University. His research interests include image/video restoration, sparse modeling, FPGA, programmable SoC, embedded vision.

Yuya KODAMA received the B.S. degree in Electrical and Electronic Engineering from Niigata University, Japan in 2020 and the M.E. degree in Electrical and Electronic Engineering from Niigata University, Japan in 2022. His research interests include signal processing, sparse modeling, FPGA, programmable SoC, embedded vision.

Hirohoshi YAMADA received the B.E., M.E., and Ph.D. degrees in electronic engineering from Hokkaido University, Sapporo, Japan, in 1988, 1990, and 1993, respectively.,Since 1993, he has been a member of the Faculty of Engineering, Niigata University, Niigata, Japan, where he is currently a Professor. From 2000 to 2001, he was a Visiting Scientist with the Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA, USA. His research interests include array signal processing, polarimetric radarinterferometry, and high-resolution techniques.

Shogo MURAMATSU (Member, APSIPA) received the B.E., M.E., and Ph.D. degrees from Tokyo Metropolitan University, Tokyo, Japan, in 1993, 1995,

and 1998, respectively. From 1997 to 1999, he was with Tokyo Metropolitan University. In 1999, he joined Niigata University, Niigata, Japan, where he is currently a Professor with the Faculty of Engineering. From 2003 to 2004, he was a Visiting Researcher with the University of Florence, Firenze, Italy. His research interests include cyber-physical systems, multidimensional signal processing, image restoration, video analysis, and embedded vision systems. Prof. Muramatsu is a senior member of the Institute of Electrical and Electronics Engineers, a senior membor of the Institute of Electronics, Information, and Communication Engineers of Japan, and a member of the Institute of Image Information and Television Engineers of Japan. He was an Associate Editor for IEEE Transactions on Signal Processing from 2019 to 2023 and was an APSIPA distinguished Lecturer from 2020 to 2021.

References

[1] A. Akhtar, B. Kathariya, and Z. Li, "Low Latency Scalable Point Cloud Communication", in *2019 IEEE International Conference on Image Processing (ICIP)*, Taipei, Taiwan: IEEE, September 2019, 2369–73, ISBN: 978-1-5386-6249-6, DOI: 10.1109/ICIP.2019.8803373, (accessed on 01/19/2022).

[2] R. Andraka, "A survey of CORDIC algorithms for FPGA based computers", in *Proceedings of the 1998 ACM/SIGDA sixth international symposium on Field programmable gate arrays*, 1998, 191–200, ISBN: 978-0-89791-978-4, DOI: 10.1145/275107.275139, (accessed on 01/19/2022).

[3] A. Avizienis, "Signed-Digit Numbe Representations for Fast Parallel Arithmetic", *IRE Transactions on Electronic Computers*, EC-10(3), 1961, 389–400, DOI: 10.1109/TEC.1961.5219227.

[4] A. Beck and M. Teboulle, "A Fast Iterative Shrinkage-Thresholding Algorithm for Linear Inverse Problems", en, *SIAM Journal on Imaging Sciences*, 2(1), January 2009, 183–202, ISSN: 1936-4954, DOI: 10.1137/080716542, (accessed on 02/07/2022).

[5] S. Boyd and L. Vandenberghe, *Convex optimization*, 15th ed., Cambridge: Cambridge university press, 2004, ISBN: 978-0-521-83378-3.

[6] T. F. Chan, J. Shen, and H.-M. Zhou, "Total Variation Wavelet Inpainting", en, *Journal of Mathematical Imaging and Vision*, 25(1), July 2006, 107–25, ISSN: 0924-9907, 1573-7683, DOI: 10.1007/s10851-006-5257-3, (accessed on 10/10/2019).

[7] Y.-K. Chen, "Challenges and opportunities of internet of things", in *17th Asia and South Pacific Design Automation Conference*, 2012, 383–8, DOI: 10.1109/ASPDAC.2012.6164978.

[8] P. Combettes and J.-C. Pesquet, "Image Restoration Subject to a Total Variation Constraint", en, *IEEE Transactions on Image Processing*, 13(9), September 2004, 1213–22, ISSN: 1057-7149, DOI: 10.1109/TIP.2004.832922, (accessed on 10/10/2019).

[9] L. Condat, "A Primal–Dual Splitting Method for Convex Optimization Involving Lipschitzian, Proximable and Linear Composite Terms", en, *Journal of Optimization Theory and Applications*, 158(2), August 2013, 460–79, ISSN: 0022-3239, 1573-2878, DOI: 10.1007/s10957-012-0245-9, (accessed on 10/10/2019).

[10] I. Daubechies, M. Defrise, and C. De Mol, "An iterative thresholding algorithm for linear inverse problems with a sparsity constraint", en, *Communications on Pure and Applied Mathematics*, 57(11), November 2004, 1413–57, ISSN: 0010-3640, 1097-0312, DOI: 10.1002/cpa.20042, (accessed on 10/10/2019).

[11] L.-J. Deng, H. Guo, and T.-Z. Huang, "A fast image recovery algorithm based on splitting deblurring and denoising", en, *Journal of Computational and Applied Mathematics*, 287 (October), October 2015, 88–97, ISSN: 03770427, DOI: 10.1016/j.cam.2015.03.035, (accessed on 02/07/2022).

[12] T. T. Duong, C. C. Pham, T. H.-P. Tran, T. P. Nguyen, and J. W. Jeon, "Near real-time ego-lane detection in highway and urban streets", en, in *2016 IEEE International Conference on Consumer Electronics-Asia (ICCE-Asia)*, Seoul: IEEE, October 2016, 1–4, ISBN: 978-1-5090-2743-9, DOI: 10.1109/ICCE-Asia.2016.7804748, (accessed on 10/10/2019).

[13] J. Faraone, M. Kumm, M. Hardieck, P. Zipf, X. Liu, D. Boland, and P. H. W. Leong, "AddNet: Deep Neural Networks Using FPGA-Optimized Multipliers", *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 28(1), 2020, 115–28, DOI: 10.1109/TVLSI.2019.2939429.

[14] M. Flynn, "Very high-speed computing systems", *Proceedings of the IEEE*, 54(12), 1966, 1901–9, DOI: 10.1109/PROC.1966.5273.

[15] R. Hu, B. S. M. R. Rao, A. Murtada, M. Alaee-Kerahroodi, and B. Ottersten, "Automotive Squint-Forward-Looking SAR: High Resolution and Early Warning", *IEEE Journal of Selected Topics in Signal Processing*, 15(4), 2021, 904–12, DOI: 10.1109/JSTSP.2021.3064175.

[16] S. Joy, M. B. S, T. B. Mukesh, M. M. Ahmed, and U. Kiran, "Real Time Road Lane Detection using Computer Vision Techniques in Python", in *2022 International Conference on Automation, Computing and Renewable Systems (ICACRS)*, 2022, 1228–32, DOI: 10.1109/ICACRS55517.2022.10029238.

[17] D. Kim and D. Park, "Element-Wise Adaptive Thresholds for Learned Iterative Shrinkage Thresholding Algorithms", *IEEE Access*, 8, 2020, 45874–86, DOI: 10.1109/ACCESS.2020.2978237.

[18] L. Kljucaric and A. D. George, "Deep Learning Inferencing with High-performance Hardware Accelerators", *ACM Trans. Intell. Syst. Technol.*, 14(4), June 2023, ISSN: 2157-6904, DOI: 10.1145/3594221, https://doi.org/10.1145/3594221.

[19] R. Kobayashi, G. Fujii, Y. Yoshida, T. Ota, F. Nin, H. Hibino, S. Choi, S. Ono, and S. Muramatsu, "Sparsity-Aware OCT Volumetric Data Restoration Using Optical Synthesis Model", *IEEE Transactions on Computational Imaging*, 8, 2022, 505–20, ISSN: 2333-9403, DOI: 10.1109/TCI.2022.3183396.

[20] T. Kobayashi, H. Yamada, Y. Yamaguchi, and Y. Sugiyama, "Simple squint angle compensation for SAR with millimeter wave automotive radar", en, in *2017 International Symposium on Antennas and Propagation (ISAP)*, Phuket: IEEE, October 2017, 1–2, ISBN: 978-1-5386-0465-6, DOI: 10.1109/ISANP.2017.8228851, (accessed on 10/10/2019).

[21] Y. Kodama, S. Muramatsu, and H. Yamada, "Fixed-Point Arithmetic of ℓ_2-Norm Approximation for 2-Tuple Arrays with Rotated ℓ_1-Norm Evaluation", in *2020 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, ISSN: 2640-0103, December 2020, 1216–21.

[22] U. Meyer-Baese, J. Chen, C. H. Chang, and A. G. Dempster, "A Comparison of Pipelined RAG-n and DA FPGA-based Multiplierless Filters", in *APCCAS 2006 - 2006 IEEE Asia Pacific Conference on Circuits and Systems*, 2006, 1555–8, DOI: 10.1109/APCCAS.2006.342540.

[23] E. Monmasson and M. N. Cirstea, "FPGA Design Methodology for Industrial Control Systems—A Review", en, *IEEE Transactions on Industrial Electronics*, 54(4), August 2007, 1824–42, ISSN: 0278-0046, DOI: 10.1109/TIE.2007.898281, (accessed on 10/10/2019).

[24] K. Naganuma and S. Ono, "Variable-Wise Diagonal Preconditioning for Primal-Dual Splitting: Design and Applications", *IEEE Transactions on Signal Processing*, 71, 2023, 3281–95, DOI: 10.1109/TSP.2023.3304789.

[25] S. Nagayama, S. Muramatsu, and H. Yamada, "Single-image Super-resolution using Complex Nonseparable Oversampled Lapped Transforms", en, in *2018 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, Honolulu, HI, USA: IEEE, November 2018, 1309–12, ISBN: 978-988-14768-5-2, DOI: 10.23919/APSIPA.2018.8659757, (accessed on 10/10/2019).

[26] N. N. Nagornov, P. A. Lyakhov, M. V. Valueva, and M. V. Bergerman, "RNS-Based FPGA Accelerators for High-Quality 3D Medical Image Wavelet Processing Using Scaled Filter Coefficients", *IEEE Access*, 10, 2022, 19215–31, DOI: 10.1109/ACCESS.2022.3151361.

[27] D. C. Nguyen, M. Ding, P. N. Pathirana, A. Seneviratne, J. Li, D. Niyato, O. Dobre, and H. V. Poor, "6G Internet of Things: A Comprehensive

Survey", *IEEE Internet of Things Journal*, 9(1), 2022, 359–83, DOI: 10.1109/JIOT.2021.3103320.

[28] S. Ono and I. Yamada, "Hierarchical Convex Optimization With Primal-Dual Splitting", en, *IEEE Transactions on Signal Processing*, 63(2), January 2015, 373–88, ISSN: 1053-587X, 1941-0476, DOI: 10.1109/TSP.2014.2373318, (accessed on 10/10/2019).

[29] L. I. Rudin, S. Osher, and E. Fatemi, "Nonlinear total variation based noise removal algorithms", en, *Physica D: Nonlinear Phenomena*, 60(1-4), November 1992, 259–68, ISSN: 01672789, DOI: 10.1016/0167-2789(92)90242-F, (accessed on 01/19/2022).

[30] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge Computing: Vision and Challenges", en, *IEEE Internet of Things Journal*, 3(5), October 2016, 637–46, ISSN: 2327-4662, DOI: 10.1109/JIOT.2016.2579198, (accessed on 10/10/2019).

[31] S. T, A. Dutt, V. Veeraiah, B. Aishwarya, K. Lal, and D. Kapila, "Analyzing The Effect of Edge Computing on Real-Time Data Processing and Latency Reduction", in *2023 10th IEEE Uttar Pradesh Section International Conference on Electrical, Electronics and Computer Engineering (UPCON)*, Vol. 10, 2023, 623–7, DOI: 10.1109/UPCON59197.2023.10434652.

[32] J. Tropp, "Greed is good: algorithmic results for sparse approximation", *IEEE Transactions on Information Theory*, 50(10), 2004, 2231–42, DOI: 10.1109/TIT.2004.834793.

[33] J. E. Volder, "The CORDIC Trigonometric Computing Technique", *IRE Transactions on Electronic Computers*, EC-8(3), September 1959, 330–4, ISSN: 0367-9950, DOI: 10.1109/TEC.1959.5222693, (accessed on 01/19/2022).

[34] J. S. Walther, "A unified algorithm for elementary functions", en, in *Proceedings of the May 18-20, 1971, spring joint computer conference on - AFIPS '71 (Spring)*, Atlantic City, New Jersey: ACM Press, 1971, 379, DOI: 10.1145/1478786.1478840, (accessed on 01/19/2022).

[35] M. Wang, Q. Wang, J. Chanussot, and D. Hong, "l0-l1 Hybrid Total Variation Regularization and its Applications on Hyperspectral Image Mixed Noise Removal and Compressed Sensing", *IEEE Transactions on Geoscience and Remote Sensing*, 59(9), 2021, 7695–710, DOI: 10.1109/TGRS.2021.3055516.

[36] S. White, "Applications of distributed arithmetic to digital signal processing: a tutorial review", *IEEE ASSP Magazine*, 6(3), 1989, 4–19, DOI: 10.1109/53.29648.

[37] J. Xiang, Y. Dong, and Y. Yang, "FISTA-Net: Learning a Fast Iterative Shrinkage Thresholding Network for Inverse Problems in Imaging", *IEEE Transactions on Medical Imaging*, 40(5), 2021, 1329–39, DOI: 10.1109/TMI.2021.3054167.

[38] H. Zhang, L. Liu, W. He, and L. Zhang, "Hyperspectral Image Denoising With Total Variation Regularization and Nonlocal Low-Rank Tensor Decomposition", *IEEE Transactions on Geoscience and Remote Sensing*, 58(5), 2020, 3071–84, DOI: 10.1109/TGRS.2019.2947333.

[39] L. Zhang, "System generator model-based FPGA design optimization and hardware co-simulation for Lorenz chaotic generator", in *2017 2nd Asia-Pacific Conference on Intelligent Robot Systems (ACIRS)*, 2017, 170–4, DOI: 10.1109/ACIRS.2017.7986087.

[40] W. Zhi-Hao, L. Siou-Wei, C. Yaw-Jong, H. Shih-Hsien, L. Yen-Cheng, and J. Gwo-Jia, "Study on vehicle safety image system optimization", en, in *2017 5th International Conference on Electrical, Electronics and Information Engineering (ICEEIE)*, Malang: IEEE, October 2017, 96–101, ISBN: 978-1-5386-0355-0, DOI: 10.1109/ICEEIE.2017.8328770, (accessed on 10/10/2019).

[41] Q. Zhu, B. Yu, Z. Wang, J. Tang, Q. A. Chen, Z. Li, X. Liu, Y. Luo, and L. Tu, "Cloud and Edge Computing for Connected and Automated Vehicles", *Foundations and Trends® in Electronic Design Automation*, 14(1-2), 2023, 1–170, ISSN: 1551-3939, DOI: 10.1561/1000000058.