mathematics

MDPI

Article

Tomographic Reconstruction: General Approach to Fast Back-Projection Algorithms

Dmitry Polevoy [1,2], Marat Gilmanov [2,3], Danil Kazimirov [2,3,*], Marina Chukalina [2,3,*], Anastasia Ingacheva [2,3], Petr Kulagin [2,3,4] and Dmitry Nikolaev [2,3]

1 Federal Research Center Computer Science and Control RAS, 119333 Moscow, Russia
2 Smart Engines Service LLC, 117312 Moscow, Russia
3 Institute for Information Transmission Problems RAS, 127051 Moscow, Russia
4 Phystech School of Applied Mathematics and Informatics, Moscow Institute of Physics and Technology (NRU), 141701 Dolgoprudny, Russia
* Correspondence: d.kazimirov@smartengines.com (D.K.); m.chukalina@smartengines.com (M.C.)

Abstract: Addressing contemporary challenges in computed tomography (CT) demands precise and efficient reconstruction. This necessitates the optimization of CT methods, particularly by improving the algorithmic efficiency of the most computationally demanding operators—forward projection and backprojection. Every measurement setup requires a unique pair of these operators. While fast algorithms for calculating forward projection operators are adaptable across various setups, they fall short in three-dimensional scanning scenarios. Hence, fast algorithms are imperative for backprojection, an integral aspect of all established reconstruction methods. This paper introduces a general method for the calculation of backprojection operators in any measurement setup. It introduces a versatile method for transposing summation-based algorithms, which rely exclusively on addition operations. The proposed approach allows for the transformation of algorithms designed for forward projection calculation into those suitable for backprojection, with the latter maintaining asymptotic algorithmic complexity. Employing this method, fast algorithms for both forward projection and backprojection have been developed for the 2D few-view parallel-beam CT as well as for the 3D cone-beam CT. The theoretically substantiated complexity values for the proposed algorithms align with their experimentally derived estimates.

Keywords: fast Hough transform; fast discrete radon transform; computed tomography

MSC: 65R32; 65R10; 68W30; 68W40; 94A08

Citation: Polevoy, D.; Gilmanov, M.; Kazimirov, D.; Chukalina, M.; Ingacheva, A.; Kulagin, P.; Nikolaev, D. Tomographic Reconstruction: General Approach to Fast Back-Projection Algorithms. *Mathematics* **2023**, *11*, 4759. https://doi.org/10.3390/math11234759

Academic Editor: Hongyu LIU

Received: 27 October 2023
Revised: 14 November 2023
Accepted: 22 November 2023
Published: 24 November 2023

1. Introduction

1.1. Problem Relevance

Computed tomography (CT) is a non-destructive X-ray technique used to probe the internal morphological structure of objects [1,2]. It plays a pivotal role in diverse fields such as medicine [3–5], industry [6–8], safety protocols [9,10], and scientific exploration, particularly in scrutinizing the morphology of promising functional materials [11]. Despite over 80 years of development in computed tomography algorithms [12,13], the pursuit of innovation and optimization remains ongoing. This is driven by various factors, including the imperative to reduce radiation dose in medical applications [14,15], augment sensitivity in screening setups [16], and refine spatial resolution in scientific [17], industrial [18], and medical [19,20] applications. Notably, CT has recently been harnessed for real-time imaging of dynamic processes [21,22]. This breakthrough is made possible only by improving classical tomographic reconstruction methods.

Modern challenges addressed by CT impose strict constraints on both reconstruction time and accuracy. Increasing spatial resolution leads to an augmented dataset, demanding enhanced computational speed for reconstruction algorithms. This goal can be pursued

through two routes: optimizing the reconstruction algorithms and/or amping up computational resources. Given the cost implications of bolstering computational power in tomographic systems, improving algorithmic efficiency stands out as the more attractive avenue.

The core of tomographic reconstruction algorithms lies in forward projection and backprojection. While both of them are linear operators, they depend on the CT setup. Various measurement setups, detailed in [23], are employed in tomograph designs. Each scheme corresponds to its specific pair of matched forward projection and backprojection operators. The scheme's parameters encompass the scanning trajectory, probing beam shape, and the dimensions and layout of the position-sensitive detector. The scanning trajectory refers to the path followed, depending on whether the object or the emitter-detector pair is stationary. The probing beam can be conical [24], fan-shaped [25], or parallel [26]. Circular [27] and helical [28] trajectories are the most commonly used for the emitter-detector pair. Some specialized tomographs utilize more intricate scanning methods [29], and non-planar detectors are utilized [30].

Calculating the forward projection and backprojection operators is considered to be the most computationally demanding aspect of tomographic reconstruction techniques. Streamlining this process holds great promise for speeding up the entire reconstruction process. The acceleration of classical algorithms of CT reconstruction is an on-going problem with main efforts being directed towards effective GPU implementations [31,32], although new fast algorithms are being developed as well [33,34]. There are fast algorithms for calculating the forward projection operator for various tomographic setups. For example, a fast approximate computation algorithm with asymptotic complexity of $\Theta\left(N^3\sqrt{N}\right)$ addition operations, where N is the linear size of the reconstruction, has been proposed for three-dimensional computer tomography [35]. Noteworthy, this algorithm can be adapted to any scanning scheme. Regrettably, these findings, on their own, do not facilitate the development of fast tomographic reconstruction methods for cone-beam and other three-dimensional scanning setups. This is due to the fact that while there are reconstruction algorithms that bypass the use of the forward projection operator (such as the Feldkamp algorithm [36]), there are no algorithms that do not employ the backprojection operator. To speed up reconstruction methods, fast algorithms for the forward projection operator, akin to those in [35], must be paired with equally fast algorithms for backprojection. These algorithms should be applicable across all scanning schemes. To address this challenge, this paper introduces a general approach to the construction of fast algorithms for calculating the transpose of operators. The proposed fast transposition algorithm efficiently handles a category of linear operators depicted as binary matrices, encompassing cases where application results rely solely on addition operations.

1.2. Related Works

Since the advent of CT, various approaches to speeding up tomographic reconstruction have emerged. They can be broadly categorized into three groups: those utilizing Fast Fourier Transform (FFT), methods employing hierarchical decomposition of the image to be recovered, and approaches applying hierarchical decomposition of linear integrals. Many researchers have proposed incorporating FFT in tomographic reconstruction algorithms. These algorithms can be fast due to projection and convolution operations in Fourier space, without completely offsetting this advantage during the transition to Fourier space. Achieving precise reconstruction through Fourier transform demands an infinite number of projections [37]. In practice, however, the available amount of projections is finite, thus the accuracy of the result depends on the approximation schemes embedded in the algorithms. As noted by Natterer [38], the presence of artifacts in reconstructed images, stemming from the Fourier transform application, reduces their practical effectiveness. Natterer delves into an analysis of the origins of these distortions in his work [38].

Various approximation schemes are tailored to maximize the accuracy of the resulting reconstruction, depending on the specific instrument configuration. In the case of paral-

lel beams, where the three-dimensional problem can be efficiently deconstructed into a series of independent two-dimensional ones, Mersereau and Oppenheim [37] advocated for reducing the approximation error by substituting the uniform polar coordinate grid in Fourier space with a grid of concentric squares. For cone beams, aiming to refine reconstruction precision, Dusaussoy [39] proposed employing a grid of concentric cubes. Averbuch et al. [40] later introduced the pseudo-polar Fourier transform for fast computation of both forward and inverse Radon transforms, highlighting Radon transform's potential in addressing parallel beam tomography challenges. In parallel schemes, an optimized Kaiser-Bessel interpolation was suggested for an algorithm utilizing non-uniform fast Fourier transform [41]. Sullivan [42] proposed improving reconstruction precision through the application of gridding methods in algorithms tailored for scenarios involving both parallel and fan beams. Subsequently, Arcadu et al. [43] further refined this algorithm.

The widely-used FDK (Feldkamp, Davis and Kress) reconstruction algorithm [36] is the go-to fast method for cone beams. Its primary limitation lies in the need for a gradual change in morphological structure along the rotational axis [39]. However, incorporating iterative strategies alongside the FDK algorithm [44] offers some flexibility in this regard. In conclusion, two key points emerge. Firstly, there is currently no universal approach to speeding up algorithms with FFT. Secondly, all known methods in this category are approximate, and their precision is a subject of scrutiny.

The following two approaches to fast reconstruction rely on hierarchical decomposition. In the first, the recoverable volume is decomposed and then the sub-volumes are aggregated. The second approach employs hierarchical decomposition of linear integrals [45,46] along with repeated use of common subsums. For example, in a parallel beam scheme [47], the square recoverable image is split into four images by vertical and horizontal lines through the center. Each of these images is reconstructed individually. This halves the linear size of the image and reduces the necessary projections by half compared to reconstructing a full-sized image, thereby reducing required operations. This approach is applied recursively. The computation speed depends on the stopping point in the hierarchical decomposition of the volume to be recovered. Reconstruction accuracy is crucially affected by the stopping point, the accuracy of interpolation used in the projection aggregation steps for reconstructing the small image, and the method used to merge the reconstructed small images. The hierarchical volume decomposition approach is not universally applicable. Different algorithms are needed and have been proposed for various measurement schemes. For a circular fan beam CT system, the algorithm is detailed in [48]. For a circular cone beam CT system, refer to [49,50]. Additionally, for a helical cone beam CT system, the algorithm is outlined in [51].

The hierarchical decomposition of linear integrals [45,46] presents a fundamentally different approach. It speeds up the computation of linear operators for both forward projection and backprojection, crucial elements in reconstruction algorithms. The key concept is based on the observation that in discrete space, lines share common subsequences of pixels, referred to as patterns, rather than intersecting at a single point. Clearly, reusing partial sums from these pattern intersections can significantly reduce the computational complexity for linear operators. To achieve fast reconstruction algorithms, it suffices to compute partial sums for a subset of possible patterns. The choice of patterns depends on the specific measurement scheme, emphasizing the need for universal transposing methods. In 1992, M. Brady and W. Yong introduced a method for fast computation of an approximate discrete Radon transform (Hough transform) on a plane, based on repeated use of common sub-sums [45]. This method, credited to Brady and Yong, appears to have been independently developed by Walter Götz, who published it in his 1993 dissertation in German. Götz's work is referenced in an English-language article by Götz and Druckmüller published in 1995 [46]. Although the Brady–Yong algorithm may not provide the most accurate approximation of straight lines by discrete patterns, it maintains a bounded approximation error. Brady and Yong's original work [45] provided an initial estimate for the deviation of position coordinates from the ideal straight line during summation. Subse-

quently, refined hypotheses were proposed [52,53], leading to the precise upper bound of the approximation error [54]. A recent algorithm, $ASD2$, has emerged for fast and accurate computation of the discrete Radon transform on a plane [55]. This approach [55] shares similarities with the Brady–Yong method but incorporates the most accurate discretization of lines (from possible discretizations), resulting in improved accuracy compared to the Brady–Yong algorithm. In 1998, T.-K. Wu and M. Brady first extended the Brady–Yong algorithm to the three-dimensional case [56]. The work [57] marked the first exploration of the precision of dyadic approximations in three-dimensional space. In 2020, a significant advancement was made by applying the Method of Four Russians, which modified the Wu–Brady technique to calculate sums over $\Theta(N^3)$ lines, rather than the excessive $\Theta(N^4)$ lines in the original Wu–Brady algorithm designed for three-dimensional tomography tasks (where N denotes the linear size of the reconstruction) [35]. This resulted in a fast algorithm for computing the forward projection operator. A notable advantage of this algorithm[35] lies in its versatility throughout CT schemes.

1.3. Contributions

This work presents a novel general method for developing efficient algorithms to compute both forward projection and backprojection linear operators. Our method transforms fast algorithms designed for forward projection, solely reliant on addition operations, into efficient algorithms for backprojection. It is noteworthy that the computational complexities for both forward projection and backprojection operators are comparable. Furthermore, this method is versatile, and applicable to computing the transpose of any operator represented by a binary matrix. The algorithm for computing the transpose of the operator exhibits a similar asymptotic computational complexity as that of the forward operator.

This work combines the Method of Four Russians with the Brady–Yong algorithm for fast computation of the forward projection operator in arbitrary measurement schemes in the 2D case. By integrating these approaches with a universal method for transposing summing algorithms, the study devises a fast algorithm for calculating backprojection operators for both parallel and fan beams within the circular CT scheme (2D case). The computational complexity of these algorithms depends on the chosen stopping point in calculating partial sums over patterns. Experimental results validate the theoretically grounded potential for fast reconstruction methods that involve computing the backprojection operator using the proposed approach of transposing algorithms.

In previous research [35], a combination of the Wu–Brady and the Method of Four Russians was utilized for the circular cone beam CT system (3D case). However, the construction of the backprojection operator was not addressed in that work. The proposed method, which transforms fast algorithms for computing forward projection operators into efficient ones for computing backprojection operators, was applied to the algorithm described in the same work [35]. An assessment of the complexity of the transposed algorithm from [35] was conducted. Implementing our method for transposing accumulative algorithms opens the door to obtaining efficient algorithms for computing backprojection operators from those for computing forward projection operators, applicable to any CT schemes in both two-dimensional and three-dimensional scenarios.

1.4. Paper Outline

The subsequent section defines the reconstruction problem and gives an overview of the employed methodologies. In the third section, we detail and justify a novel method for transposing summing algorithms. This method allows for deriving an algorithm to compute the transpose of an operator represented by a binary matrix through the original operator computation algorithm. The evaluation of the algorithmic complexity of the forward and transposed operators computation is presented within Section 3.3. In the fourth section, we apply this transposition method to established fast algorithms for computing forward projection operators in both 2D and 3D scenarios. Section 4 introduces novel fast algorithms for computing both forward projection and backprojection operators based on the fast

Hough transform (FHT), accompanied by a complexity assessment. The 2D case is dealt with in Section 4.2 and the 3D case is the subject of Section 4.4. These algorithms are derived through the application of our proposed method for transposing algorithms, and a thorough complexity analysis is performed. The fifth section delves into the experimental validation of our proposed method of algorithm transposition. Finally, the paper concludes with a discussion highlighting the advantages of the proposed approach.

2. CT Reconstruction: Problem Statement and Solutions Overview

CT reconstruction entails creating a digital representation of an object based on a collection of measured X-ray images or tomographic projections. This digital depiction represents a discrete spatial distribution of the linear attenuation coefficient of X-ray radiation.

2.1. Projection Image Model in Parallel-Beam Circular CT

The classical mathematical model for generating a CT projection image is based on the following assumptions:

- Monochromatic radiation is used for scanning;
- The imaging process is approximated with geometric optics;
- There are no radiation sources (primary or secondary) within the examined sample;
- Radiation attenuation depends on the absorptive properties of the material and follows the Bouguer–Beer–Lambert law;
- the detector only records the attenuated probing radiation. The detector cell has an infinitesimal size, a linear response function, and unit sensitivity.

The basic scheme for measuring projections with a parallel beam on an object is illustrated in Figure 1.

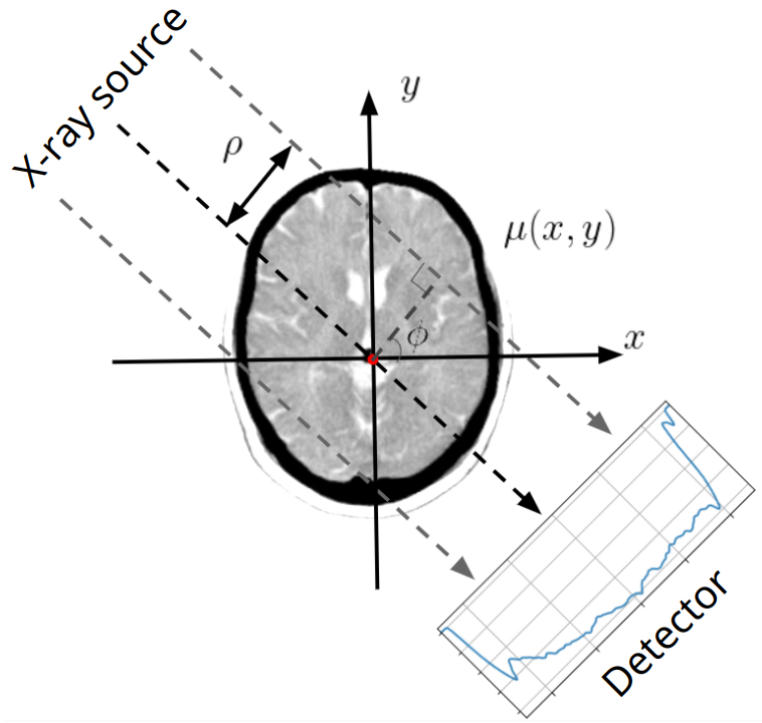

Figure 1. A schematic illustration of parallel-beam CT. Red point in the center illustrates the rotation axis, dashed lines are the X-ray propagation directions, ϕ is the rotation angle, ρ is detector cell coordinate and the blue line is the effective signal registered by detector.

When an object is probed with a parallel beam and the axis of rotation is perpendicular to the beam, we can analyze the CT problem on a plane without loss of generality. Consider

a parallel set of planes perpendicular to the rotation axis. Measurements in each of these regions are independent. Let us take advantage of this and focus on a single plane.

The Cartesian coordinate system $x0y$ in the plane aligns with the center of rotation of the source-detector system, and the spatial distribution of the linear attenuation coefficient $\mu(x, y)$ of the probing radiation is described by a finite function. In this coordinate system, we will use the standard parameterization of a line: (ρ, φ). All rays in the probing beam at one projection position share the same φ coordinate. The image is captured by a planar position-sensitive detector located perpendicular to the probing direction. The ρ coordinate precisely locates the detector cell. For an infinitely thin X-ray beam at an angle φ reaching the detector at ρ, the radiation attenuation law through the object is expressed as:

$$I(\rho, \varphi) = I_0 \exp\left(- \int_{-\infty}^{\infty} \mu(\rho \cos\varphi - l \sin\varphi, \rho \sin\varphi + l \cos\varphi)\, dl \right), \tag{1}$$

where I_0 is the source's radiant exitance, and I is the irradiance of the detector cell. Normalizing by I_0 and taking the logarithm yields the function $g(\rho, \phi)$, referred to as the ray integral:

$$g(\rho, \phi) = -\ln \frac{I(\rho, \phi)}{I_0} = \int_{-\infty}^{\infty} \mu(\rho \cos\phi - l \sin\phi, \rho \sin\phi + l \cos\phi)\, dl. \tag{2}$$

Through tomographic measurements, we acquire linear integrals of the function $\mu(x, y)$ along each straight line ℓ. This process, known as the Radon transform, maps a function from Euclidean space to its set of linear integrals. Radon introduced an explicit inversion formula for an infinite set of straight lines in 1917 [58]. Building on this, the problem of tomographic reconstruction involves recovering the function $\mu(x, y)$ when linear integrals are known for a finite set of straight lines. The arrangement of these lines is referred to as the measurement scheme, which is determined by the tomograph's design, including the shape of the probing beam and the scanning scheme. Consequently, the reconstruction process must align with the chosen measurement scheme.

In practice, synchrotron radiation sources primarily employ monochromatic probing. Medical and industrial tomographs, on the other hand, utilize polychromatic radiation. Different approaches have been suggested to linearize the relationship between measurement results and the function describing attenuation. These methods vary from those needing extra calibration measurements [59,60] to fully automated techniques [61,62]. Note that using parallel beams for scanning is primarily a feature of synchrotrons. In laboratory settings, achieving a parallel scheme necessitates the incorporation of extra optical components in the tomograph setup. Conversely, when employing an X-ray tube to generate the beam without extra optical elements, a cone beam is formed. In this setup, the source is situated at the apex of the cone, while the detector rests in the plane of its base. The object being probed is located within the cone. The collection of rays composing a tomographic projection under the current angle in a cone scheme differs from that in a parallel scheme for the same angle.

Consider a circular scanning path. In tomographic research, measurements are taken along a specified finite number of directions C, which determine the set of projection angles $\Phi = \{\phi_i\}_{i=1}^{C}$. For each projection angle, the linearized measurements are recorded as a vector $\overline{g}(\phi) = (g_1, \ldots, g_N)^T$, where N represents the number of cells in the detector array (in the two-dimensional scenario). The measurements for all angles can be consolidated into a single vector $\overline{G} = (\overline{g}(\phi_1), \ldots, \overline{g}(\phi_C))^T = (g_i : i \in \mathbb{Z}_{1,N \cdot C})^T$. The set of rays constituting the vector for the current projection angle in a cone beam differs from the set of rays forming the vector in a parallel beam for the same angle.

To perform reconstruction, we need to produce measured data discretization first. For μ discretization, the reconstructed area is divided into a regular grid of square pixels. The number of pixels in the horizontal direction is the same as in the vertical direction, and it

matches the number of detector cells N. This grid contains a total of N^2 elements, which can be expressed as a vector $\overline{M} = (\mu_{1,1}, \ldots, \mu_{1,N}, \mu_{2,1}, \ldots, \mu_{N,N})^T = (\mu_i : i \in \mathbb{Z}_{1,N^2})^T$.

The vector $\overline{M}$ represents a 2D slice of the reconstructed 3D image. Ray integral values (2) are approximated using sums. The model for generating projection data is expressed in matrix form:

$$\mathbb{W}\overline{M} = \overline{G}, \tag{3}$$

where $\mathbb{W}$ is the projection matrix of size $N \cdot C \times N^2$. The elements $0 \leq w_{i,j} \leq 1$ in this matrix specify each pixel's contribution to the sum. This $\mathbb{W}$ matrix defines the measurement setup.

The dimensions of the projection matrix depend on the number of detector cells and the quantity of projection angles. Its configuration is influenced by factors like scanning path, probe shape, the number of projection angles, and their absolute values.

2.2. CT Reconstruction Algorithms

In the formulation (3), the CT problem involves reconstructing a digital image of the studied object $\overline{M}$ given the known projection matrix $\mathbb{W}$ and the vector of linearized measured values $\overline{G}$.

The matrix $\mathbb{W}$ is a sparse non-negative matrix, with elements representing weights calculated based on the chosen ray-pixel intersection model [23]. We will treat the projection matrix $\mathbb{W}$ as a binary matrix for the forward projection operator in the natural basis of the introduced Cartesian coordinate system $x0y$. The value $w_{i,j}$ of the projection matrix element at indices i, j equals 1 if and only if the X-ray beam intersects the pixel of the reconstructed object's cross-section at coordinates i, j.

Various approaches have been proposed for reconstructing images from measured projections. Integral methods rely on the numerical implementation of the Radon transform inversion formulas. The most widely used method, found in most medical tomographs, is the convolution and backprojection technique.

2.3. CT Reconstruction Algorithms for Parallel Scanning Scheme

2.3.1. Convolution and Back-Projection Algorithms (Filtered BackProjection, FBP)

In tomography, a widely used family of algorithms for reconstructing images is based on the convolution and backprojection method, known as Filtered Back-Projection (FBP) [63]. Buzug et al. [23] demonstrated that with the matrix representation from (3), the following relationship can be derived:

$$\overline{M} = (\mathbb{W}^T \cdot \mathbb{W})^{-1} \cdot \mathbb{W}^T \cdot \overline{G} = \mathbb{W}^T \cdot (\mathbb{W} \cdot \mathbb{W}^T)^{-1} \cdot \overline{G}, \tag{4}$$

where $\mathbb{W}^T$ represents the backprojection operator matrix. The matrices $(\mathbb{W}^T \cdot \mathbb{W})^{-1}$ and $(\mathbb{W} \cdot \mathbb{W}^T)^{-1}$ are filtering operators that can be applied either after or before the projection, respectively.

The convolution and backprojection method is a fast two-step approach. Initially, measured projections are convolved with a filter [64]. Subsequently, a backprojection operation is conducted. Proposed optimizations targeted hardware solutions. Myagotin et al. presented an implementation of the method on multi-core processors [65]. The method operates with low memory requirements due to its sequential execution. However, it requires a substantial number of projections. To ensure accurate reconstruction, projection angles should be evenly distributed within the 0 to 180-degree range, with the number of angles meeting the Nyquist criterion. Otherwise, additional data interpolation may be necessary [66], potentially impacting reconstruction accuracy. In cases of limited projections, an algebraic reconstruction approach has demonstrated effectiveness.

2.3.2. Algebraic Reconstruction: SIRT (Simultaneous Iterative Reconstruction Technique)

The algebraic approach to CT reconstruction, used alongside FBP [67], treats (3) as a system of linear algebraic equations, seeking a solution for $\overline{M}$. Due to the system's high dimensionality, direct methods for solving linear systems of equations are not feasible. Specialized algorithms have been proposed for this purpose [68,69]. In contrast to the algebraic reconstruction techniques proposed by Gordon et al. [68], where solution vector values are updated sequentially, the SIRT method, introduced by Gilbert in 1972 [69], updates all coordinates of the solution vector at each step. For each coordinate, the current value is calculated, taking into account all registered contributions. The solution at each iteration in matrix form is updated as follows:

$$\overline{M}^{(p+1)} = \overline{M}^{(p)} + \lambda_p \cdot \mathbb{R} \cdot \mathbb{W}^T \cdot \mathbb{Q} \cdot (\overline{G} - \mathbb{W} \cdot \overline{M}^{(p)}), \tag{5}$$

where λ_p is the relaxation parameter [70] and $\mathbb{W}^T$ represents the matrix of the backprojection operator. The diagonal matrices $\mathbb{R}$ and $\mathbb{Q}$ are of dimensions $N^2 \times N^2$ and $N \cdot C \times N \cdot C$, respectively. Their elements are computed using the following formulas: $r_{ii} = 1/\sum_j w_{ij}$ and $q_{jj} = 1/\sum_i w_{ij}$. When the number of projection angles and/or the angle range are restricted, employing the Total Variation (TV) method [71] for regularization yields a highly precise result. In such cases, the reconstruction is acquired by solving the convex optimization problem:

$$\|\mathbb{W}\overline{M} - \overline{G}\| + \alpha\|\overline{M}\|_{TV} \rightarrow \min_{\overline{M}}, \tag{6}$$

where α is the regularization parameter [71]. The optimized expression, and thus the iteratively obtained solution, includes both forward projection and backprojection operations. While the algebraic approach to CT reconstruction demands more computational resources compared to the convolution and backprojection method, its significant advantage lies in its lack of strict limitations on the number and quality of tomographic projections, as well as its independence from uniform distribution of observed projection angles. Fast reconstruction using iterative algorithms can be achieved through two approaches. The first involves augmenting computational resources and devising methods for parallel processing [72]. The second focuses on designing and utilizing algorithms with reduced computational complexity. While given specific measurement schemes, there are efficient algorithms for forward projection operator computation [52], and fast computation of the backprojection operator is not guaranteed.

2.3.3. Neural Network-Based Reconstruction Techniques

The development of neural network-based approaches to tomographic reconstruction began in 1995 with the landmark work by Kerr et al. [73]. They demonstrated that a neural network could produce tomographic images of comparable accuracy to those used in its training. This was based on the neural network's capacity to discern and evaluate intricate functional relationships. The neural network was trained using reconstructions obtained through the FBP method. Nowadays, the integration of neural network models into reconstruction algorithms facilitates high-quality reconstructions in few-view CT [74], especially in cases when the full range of angles is limited [75,76], or when the measured projections are noisy [77]. The method's groundbreaking applications, such as tomographic measurements with nanometer resolution, real-time CT, and studying dynamic processes, pose challenges for traditional methods. In neural network approaches, the operations of forward projection and backprojection remain the mathematical cornerstone.

2.4. Fast Approximation Schemes

Efficient CT reconstruction algorithms can be built with fast approximate computational schemes that utilize the Fast Hough Transform (FHT), also known as the Fast Discrete Radon Transform [45,78–81]. The Brady–Yong algorithm [45] approximates straight lines

using discrete dyadic patterns. While these patterns enable optimized summation calculations, they exhibit some degree of approximation inaccuracy. For comprehensive insights into the general properties and practical applications of FHT in 2D cases, refer to [82,83].

In [84], FHT was proposed for speeding up the Simultaneous Algebraic Reconstruction Technique (SART) [85] in 2D reconstruction. The authors applied FHT for both forward projection operator calculation of residuals and for transposed operator calculation of corrections in the iterative scheme. This led to a reduction in the asymptotic complexity of a single iteration from $\Theta(N^3)$ to $\Theta(N^2 \log N)$. Additionally, in [86], a modification of the FBP algorithm, which employs FHT to speed up the most computationally intensive step (backprojection) is introduced. This modification allows for the image reconstruction with $\Theta(N^2 \log N)$ addition operations and $\Theta(N^2)$ multiplication operations.

In 3D CT reconstruction schemes, ref. [35] demonstrated that summing over $\Theta(N^3)$ patterns can be computed in $\Theta(N^{7/2})$ addition operations by utilizing a generalization of FHT for 3D introduced by Ershov et al. [87]. However, ref. [35] provides only a theoretical estimate and a general outline of the algorithm without a specific implementation, and the algorithm for the fast computation of the transposed operator was not discussed.

Thus, reducing the constant in the computation complexity of the forward projection algorithm and devising fast schemes for the transposed operator will substantially improve the efficiency of various CT reconstruction algorithms.

3. Transposing the Summation Algorithm for Computing the Summation Operator, Represented as a Product of Boolean Matrices

3.1. Notations and Definitions

Let $\mathbb{B}$ represent a Boolean matrix defined as

$$\mathbb{B} \stackrel{\text{def}}{=} (b_{y,x} \in \{0,1\} : y \in \mathbb{Z}_{0,H-1}, x \in \mathbb{Z}_{0,W-1}), \tag{7}$$

where y denotes the index of the row in matrix $\mathbb{B}$, x represents the index of the column in matrix $\mathbb{B}$, $H \equiv H(\mathbb{B})$ signifies the number of matrix rows, and $W \equiv W(\mathbb{B})$ indicates the number of matrix columns, $H(\mathbb{B}), W(\mathbb{B}) \in \mathbb{Z}_{1,\infty}$.

The number of ones in matrix $\mathbb{B}$ is denoted as $C_{\mathbb{1}}(\mathbb{B})$ and can be computed using the formula:

$$C_{\mathbb{1}}(\mathbb{B}) \stackrel{\text{def}}{=} \sum_{y,x} b_{y,x}. \tag{8}$$

The count of summations for a Boolean matrix $\mathbb{B}$ is denoted as $C_{Summ}(\mathbb{B})$ and is calculated as follows:

$$C_{Summ}(\mathbb{B}) \stackrel{\text{def}}{=} C_{\mathbb{1}}(\mathbb{B}) - H(\mathbb{B}). \tag{9}$$

The number of duplications for a Boolean matrix $\mathbb{B}$ is denoted as $C_{Copy}(\mathbb{B})$ and is determined as

$$C_{Copy}(\mathbb{B}) \stackrel{\text{def}}{=} C_{\mathbb{1}}(\mathbb{B}) - W(\mathbb{B}). \tag{10}$$

Let $dec^p(\cdot)$ represent a predefined decomposition of a Boolean matrix into the product of p Boolean matrices:

$$dec^p(\mathbb{B}) \stackrel{\text{def}}{=} \prod_{i=p}^{1} dec_i^p(\mathbb{B}) = \prod_{i=p}^{1} \mathbb{B}_i = \mathbb{B}_p \mathbb{B}_{p-1} \ldots \mathbb{B}_1 = \mathbb{B}, \tag{11}$$

where $dec_i^p(\mathbb{B}) \stackrel{\text{def}}{=} \mathbb{B}_i$ is the i-th component of the decomposition. In general, we assume $dec^1(\mathbb{B}) \equiv \mathbb{B}$.

Similar related functions $C_{\mathbb{1}}(dec^p(\cdot))$, $H(dec^p(\cdot))$, $W(dec^p(\cdot))$, $C_{Summ}(dec^p(\cdot))$ and $C_{Copy}(dec^p(\cdot))$ for the decomposition of a Boolean matrix into the product of p Boolean

matrices are computed by summing the corresponding function over all matrices in the decomposition:

$$C_{\mathbb{1}}(dec^p(\mathbb{B})) \stackrel{\text{def}}{=} \sum_{i=1}^{p} C_{\mathbb{1}}(dec_i^p(\mathbb{B})), \tag{12}$$

$$H(dec^p(\mathbb{B})) \stackrel{\text{def}}{=} \sum_{i=1}^{p} H(dec_i^p(\mathbb{B})), \tag{13}$$

$$W(dec^p(\mathbb{B})) \stackrel{\text{def}}{=} \sum_{i=1}^{p} W(dec_i^p(\mathbb{B})), \tag{14}$$

$$\begin{gathered} C_{Summ}(dec^p(\mathbb{B})) \stackrel{\text{def}}{=} \sum_{i=1}^{p} C_{Summ}(dec_i^p(\mathbb{B})) = \sum_{i=1}^{p} [C_{\mathbb{1}}(dec_i^p(\mathbb{B})) - H(dec_i^p(\mathbb{B}))] = \\ C_{\mathbb{1}}(dec^p(\mathbb{B})) - H(dec^p(\mathbb{B})), \end{gathered} \tag{15}$$

$$C_{Copy}(dec^p(\mathbb{B})) \stackrel{\text{def}}{=} \sum_{i=1}^{p} C_{Copy}(dec_i^p(\mathbb{B})) = C_{\mathbb{1}}(dec^p(\mathbb{B})) - W(dec^p(\mathbb{B})). \tag{16}$$

3.2. Correspondence of Decompositions for Forward and Transposed Operators in Boolean Matrix Products

Let us consider the approximation of ray sum values in matrix form as shown in Equation (3). We use a linear operator:

$$\mathcal{B} : \mathbb{R}^n \to \mathbb{R}^m. \tag{17}$$

The matrix $\mathbb{B}$ of this operator is Boolean within the previously introduced Cartesian coordinate system $x0y$ ($n, m \in \mathbb{Z}_{1,\infty}$). Without loss of generality, we further assume that the matrix $\mathbb{B}$ of the operator $\mathcal{B}$ and all matrices $dec_i^p(\mathbb{B})$ in the decomposition do not contain zero rows and columns. This is because, if they did, the algorithmic complexity of computing $\mathcal{B}\overline{M}$ could be reduced by excluding a portion of the input data $\overline{M}$ from consideration, where $\overline{M} = (\mu_i \in \mathbb{R} : i \in \mathbb{Z}_{1,n})^T$.

The transposed operator (17) of $\mathcal{B}$, denoted as $\mathcal{B}^T$, defined by the matrix $\mathbb{B}^T$:

$$\mathcal{B}^T : \mathbb{R}^m \to \mathbb{R}^n. \tag{18}$$

For every decomposition $dec^p(\mathbb{B})$ of the matrix $\mathbb{B}$ representing the forward operator $\mathcal{B}$, there is a corresponding decomposition $dec^p(\mathbb{B}^T)$ of the matrix $\mathbb{B}^T$ for the transposed operator $\mathcal{B}^T$. This is computed by considering the transposition of matrix products:

$$dec^p(\mathbb{B}^T) = \mathbb{B}^T = (\mathbb{B}_p\mathbb{B}_{p-1}\ldots\mathbb{B}_1)^T = \mathbb{B}_1^T\ldots\mathbb{B}_{p-1}^T\mathbb{B}_p^T = \prod_{i=p}^{1}\mathbb{B}_{p-i+1}^T. \tag{19}$$

3.3. Summation-Based Algorithms and Matrix Decompositions for Forward and Transposed Operators as the Product of Boolean Matrices—Algorithmic Complexity in Computing Forward and Transposed Operators

Let us label an algorithm as summation-based if it relies solely on addition operations for data manipulation.

The algorithm for computing the result $\mathcal{B}\overline{M}$ after applying the operator $\mathcal{B}$ defined as (17) to the input vector $\overline{M}$, expressed as

$$\mathcal{B}\overline{M} = dec^p(\mathbb{B})\overline{M}, \tag{20}$$

employs a sequential (row-wise) multiplication of matrix components $dec_i^p(\mathbb{B}) \equiv \mathbb{B}_i, i \in \mathbb{Z}_{1,p}$, in the decomposition $dec^p(\mathbb{B})$ of the operator's matrix $\mathbb{B}$ with the input vector $\overline{M}$. This process employs only summation, hence its algorithmic complexity is $C_{Summ}(dec^p(\mathbb{B}))$. In this algorithm, the j-th component of the vector $\overline{V}^{(i)} = \mathbb{B}_i\overline{V}^{(i-1)}$, $i \in \mathbb{Z}_{1,p}$ ($\overline{V}^{(0)} = \overline{M}$, $\overline{V}^{(p)} = \mathcal{B}\overline{M}$), is computed as the product of the j-th row of the matrix $\mathbb{B}_i$ and the column vector $\overline{V}^{(i-1)}$. This computation requires one less addition operation than the number of non-zero elements in the j-th row of the matrix $\mathbb{B}_i$. Consequently, to compute all components of the vector $\overline{V}^{(i)} = \mathbb{B}_i\overline{V}^{(i-1)}$, $C_{\mathbb{1}}(\mathbb{B}_i) - H(\mathbb{B}_i) \equiv C_{Summ}(\mathbb{B}_i), i \in \mathbb{Z}_{1,p}$ addition operations are performed. In total, the algorithm requires $\sum\limits_{i=1}^{p} C_{Summ}(\mathbb{B}_i) \equiv C_{Summ}(dec^p(\mathbb{B}))$ addition operations.

In a similar manner, the algorithm for computing the result of applying the transposed operator $\mathcal{B}^T$ to the input vector, achieved through sequential (row-wise) multiplication of matrix components $dec_i^p(\mathbb{B}^T) \equiv \mathbb{B}^T_{p-i+1}, i \in \mathbb{Z}_{1,p}$, in the decomposition $dec^p(\mathbb{B}^T)$ of the matrix $\mathbb{B}^T$ for the operator $\mathcal{B}^T$, is also summation-based only with an algorithmic complexity of $C_{Summ}(dec^p(\mathbb{B}^T))$.

Now, let us determine the complexity relationship between the algorithms for computing the results of applying the forward and transposed operators when utilizing a pair of corresponding decompositions:

$$\begin{gathered} C_{Summ}(dec^p(\mathbb{B})) - C_{Summ}(dec^p(\mathbb{B}^T)) = \\ \sum_{i=1}^{p}(C_{\mathbb{1}}(dec_i^p(\mathbb{B})) - H(dec_i^p(\mathbb{B}))) - \sum_{i=1}^{p}(C_{\mathbb{1}}(dec_i^p(\mathbb{B}^T)) - H(dec_i^p(\mathbb{B}^T))) = \\ \sum_{i=1}^{p}(C_{\mathbb{1}}(dec_i^p(\mathbb{B})) - C_{\mathbb{1}}(dec_i^p(\mathbb{B}^T))) + \sum_{i=1}^{p}(H(dec_i^p(\mathbb{B}^T)) - H(dec_i^p(\mathbb{B}))). \end{gathered} \tag{21}$$

Since the number of non-zero elements in the corresponding matrix components of the decompositions $dec^p(\mathbb{B})$ and $dec^p(\mathbb{B}^T)$ for the matrices $\mathbb{B}$ and $\mathbb{B}^T$ of the forward and transposed operators $\mathcal{B}$ and $\mathcal{B}^T$ is the same (i.e., $C_{\mathbb{1}}(dec^p(\mathbb{B})) = C_{\mathbb{1}}(dec^p(\mathbb{B}^{\mathbb{T}}))$), and for the decomposition $dec^p(\mathbb{B})$, the relationships $H(dec_i^p(\mathbb{B})) = W(dec_{i+1}^p(\mathbb{B}))$, $i \in \mathbb{Z}_{1,p-1}$, hold due to the compatibility of the dimensions of the multiplied matrix components $\mathbb{B}_j$, $j \in \mathbb{Z}_{1,p}$ in the decomposition $dec^p(\mathbb{B})$, equality (21) can be extended:

$$\begin{gathered} \sum_{i=1}^{p}(C_{\mathbb{1}}(dec_i^p(\mathbb{B})) - C_{\mathbb{1}}(dec_i^p(\mathbb{B}^{\mathbb{T}}))) + \sum_{i=1}^{p}(H(dec_i^p(\mathbb{B}^{\mathbb{T}})) - H(dec_i^p(\mathbb{B}))) = \\ \sum_{i=1}^{p}(H(dec_i^p(\mathbb{B}^{\mathbb{T}})) - H(dec_i^p(\mathbb{B}))) = H(dec_p^p(\mathbb{B}^T)) - H(dec_p^p(\mathbb{B})) = \\ W(dec_1^p(\mathbb{B})) - H(dec_p^p(\mathbb{B})) = W(\mathbb{B}) - H(\mathbb{B}). \end{gathered} \tag{22}$$

Combining (21) and (22), we derive the following:

$$C_{Summ}(dec^p(\mathbb{B})) - C_{Summ}(dec^p(\mathbb{B}^T)) = W(\mathbb{B}) - H(\mathbb{B}) = n - m. \tag{23}$$

Hence, the difference in algorithmic complexities when computing the results of applying forward and transposed operators using corresponding Boolean decompositions is a constant determined by the dimensions of the input and output vectors. The former result is illustrated in Figure 2.

Given that the precise computation of forward or transposed operators requires no less than n addition operations, the formula (23) implies that if $n - m(n) = \Theta(n)$ as $n \to \infty$, then $C_{Summ}(dec^p(\mathbb{B}^T)) = \Theta(C_{Summ}(dec^p(\mathbb{B})))$ and $C_{Summ}(dec^p(\mathbb{B})) = \Theta(C_{Summ}(dec^p(\mathbb{B}^T)))$ as $n \to \infty$. In other words, the asymptotic algorithmic complexities for the precise computation of forward and transposed operators are of the same order.

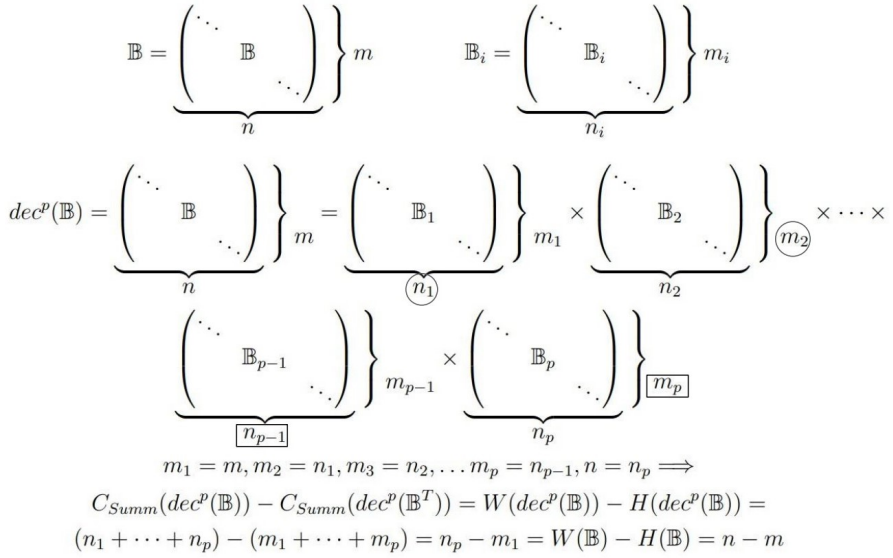

Figure 2. Relations between dimensions of Boolean decomposition factors and the count of summations of forward and transposed decompositions.

3.4. Universal Method for Deriving Efficient Transposed Operator Calculation Algorithms

Consider a Boolean matrix $\mathbb{B}$ representing the linear operator $\mathcal{B}$, which can be decomposed into the product of Boolean matrices as $dec^p(\mathbb{B}) = \mathbb{B}_p\mathbb{B}_{p-1}\ldots\mathbb{B}_1$, which aligns with established fast algorithm for computing the operator $\mathcal{B}$. Subsequently, the decomposition derived from $dec^p(\mathbb{B})$, denoted as $dec^p(\mathbb{B}^T) = \mathbb{B}_1^T\mathbb{B}_2^T\ldots\mathbb{B}_p^T$ or the matrix $\mathbb{B}^T$ of transposed operator $\mathcal{B}^T$, leads to an efficient algorithm for computing $\mathcal{B}^T$. The algorithmic complexity is asymptotically of the same order as that of computing the original operator $\mathcal{B}$.

The described method is a versatile approach for transposing the algorithm used to compute the forward operator. This can be implemented in the form of sequential multiplications of Boolean matrices, which are derived from the known computationally efficient forward operator matrix decomposition, with an input data vector. It enables the derivation of an algorithm for computing the transposed operator with a well-defined asymptotic complexity, which is on par with that of computing the forward operator.

Next, we will explore the practical implementations of this generalized transposition method in the context of summation-based algorithms for computing backprojection operators in CT reconstruction.

4. Inverting the Projection Operator Calculation Algorithm (FHT Approximation)

4.1. Fundamental 2D FHT Concepts

Consider a standard rectangular Cartesian coordinate system denoted as $X0Y$ associated with an image. This image can be represented as a $2^n \times 2^n, n \in \mathbb{Z}_{1,\infty}$ matrix, I_{2^n}. The bottom-left corner of the image corresponds to the point 0, and each element of the image, known as a pixel, is depicted as a square with a side length of 1 (see Figure 3).

Let us consider the parametrization (s, t) of straight lines on the plane $X0Y$ [81]. This defines a continuous straight line passing through points on the lines that bound the image (refer to Figure 3):

- A predominantly vertical straight line (PVL) passes through points $(s_C, 0)$ and $(s_C + t_C, 2^n)$, situated on the straight lines that bound the image from above and below;
- A predominantly horizontal straight line (PHL) passes through points $(0, s_C)$ and $(2^n, s_C + t_C)$, situated on the straight lines that bound the image from the left and right.

For simplicity (similar to [45]), we focus exclusively on predominantly vertical straight lines with a negative slope $t \leq 0$.

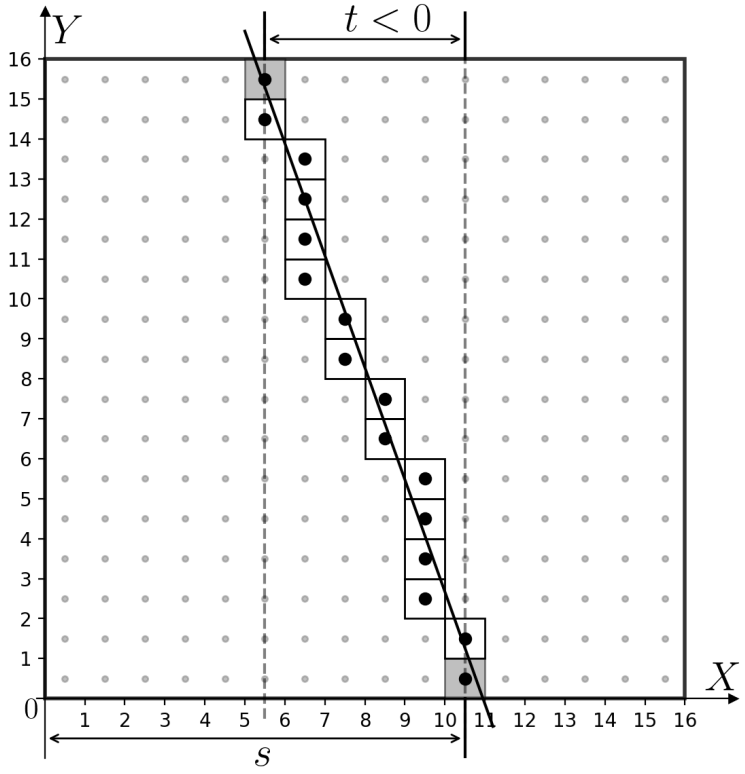

Figure 3. Illustration of $X0Y$ axes, I_{2^4}, and the structure of the dyadic pattern approximating a straight line ($s = 10$, $t = -5$). The black dots indicate the centers of the dyadic pattern pixels.

The discrete set

$$\Omega = \mathbb{Z}_{0,2^n-1} \times \mathbb{Z}_{0,2^n-1} \tag{24}$$

defines the coordinates of the pixels $(x, y) \in \Omega$. A discrete straight line (straight-line pattern) is a subset of Ω that approximates a continuous straight line. It is defined by a pair of pixels with coordinates $(s_D, 0) \in \Omega$ and $(s_D - t_D, 2^n - 1) \in \Omega$ along the image boundaries. Here, $s_D \in \mathbb{Z}_{0,2^n-1}$ represents the integer shift at the pattern's beginning, and $t_D \in \mathbb{Z}_{0,2^n-1}$ denotes the integer shift of the pattern's end relative to its beginning (see Figure 3). The continuous straight lines extending beyond the image are approximated by a straight-line pattern defined as the pattern for the original continuous straight line within the region of the extended zero-padded image.

The creators of the FHT algorithm [45] proposed using a dyadic pattern as a discrete straight line, which can be defined both recursively [45] and analytically [53]. The properties of this pattern, including estimates of its maximum deviation from a continuous straight line, have been explored in [46,53,54].

4.2. Two-Dimensional Operator Computation Algorithms

In the development of CT reconstruction algorithms, particularly in cases with a small number of angles (few-view CT) [88] and/or monitored reconstruction (monitored CT) [89], the process of summing over all directions may prove to be excessive (refer to the analysis in Section 4.2.5). Hence, we will now explore a more adaptable approach. This entails computing sums for patterns of length $2^k, k < n$, followed by aggregating these partial sums to derive totals along all discrete straight lines in the specified set of directions.

4.2.1. Two-Dimensional Projection Algorithm with FHT Termination and Aggregation (for Arbitrary Set of Straight Lines)

The original Brady–Yong algorithm [45] calculates sums for all discrete straight lines passing through the image. The algorithm *dirPFHT* (Directed Partial Fast Hough Trans-

form) is a modification of the Brady–Yong algorithm. For an input image I_{2^n} with dimensions $2^n \times 2^n$, it computes the output vector of sums $\bar{s} = (s_j)_{j=0}^{q-1}$ for q discrete straight lines parameterized in (s,t), defined as a set $L = \{(x_j, a_j)\}_{j=0}^{q-1}$.

Main Computation Stages:

1. (Right) zero padding the original image I_{2^n} to R_0;
2. Computing partial sums along patterns of length 2^k (using *dirPFHTk*);
3. Aggregating sums for each discrete straight line from L (using *dirSUMk*).

Algorithm 1 provides pseudocode for two-dimensional projection algorithm with termination on kth iteration and aggregation.

Algorithm 1 Algorithm for forward FHT with termination and aggregation

procedure $dirPFHT(L, I_{2^n}, n, k)$
  $R_0(x,y,0) \leftarrow I_{2^n}(x,y) \quad \forall x \in \mathbb{Z}_{0,2^n-1}, y \in \mathbb{Z}_{0,2^n-1}$ ▷ copy the original data
  $R_0(x,y,0) \leftarrow 0 \quad \forall x \in \mathbb{Z}_{2^n,2^{n+1}-1}, y \in \mathbb{Z}_{0,2^n-1}$ ▷ (right) zero padding
  $R_k \leftarrow dirPFHTk(R_k, n, k)$ ▷ compute sums along patterns with length 2^k
  $s_j \leftarrow dirSUMk(x_j, a_j, R_k, n, k) \quad \forall j \in \mathbb{Z}_{0,q-1}, (x_j, a_j) \in L$ ▷ calculate sums along discrete straight lines from L
  return s

Algorithm 2 *dirPFHTk* computes and stores the sums for the input tensor R_0 in the form of a tensor R_k, summing over sub-patterns of length 2^k. When $k = n$, the result conforms to the output of the Brady algorithm [45].

Algorithm 2 Forward FHT with termination

procedure $dirPFHTk(R_0, n, k)$
  for $i = 1$ to k **do** ▷ along the exponent range of the pattern length
    for $a = 0$ to $2^i - 1$ **do** ▷ along the absolute $0X$ shift of the pattern end
      for $y = 0$ to $2^n - 2^i$ step 2^i **do** ▷ along the $0Y$ shift of the pattern beginning
        for $x = 0$ to $2^{n+1} - 1$ **do** ▷ along the $0X$ shift of the pattern beginning
          $x_2 \leftarrow (x - \lceil a/2 \rceil) \mod 2^{n+1}$ ▷ circular shift
          $y_2 \leftarrow y + 2^{i-1}$
          $R_i(x,y,a) \leftarrow R_{i-1}(x,y,\lfloor a/2 \rfloor) + R_{i-1}(x_2,y_2,\lfloor a/2 \rfloor)$ ▷ saving the sum of two sub-patterns
  return R_k

Algorithm 3 *dirSUMk* aggregates the sum along a pattern with specified parameters (x,a), where $x \in \mathbb{Z}_{0,2^n-1}, a \in \mathbb{Z}_{0,2^n-1}$, for the tensor R_k with sums along sub-patterns of length 2^k. It does so through a recursive call to *recSUMk*.

Algorithm 3 Aggregation for the forward projection operator

procedure $dirSUMk(x, a, R_k, n, k)$
  return $recSUMk(x, 0, a, n, R_k, n, k)$

The Algorithm 4 *recSUMk* recursively aggregates the precomputed partial sums over patterns of length $2^i, i \in \mathbb{Z}_{k,n}$, in R_k along a discrete straight line given by (x,a).

Algorithm 4 Aggregation

procedure $recSUMk(x, y, a, i, R_k, n, k)$
  if $i = k$ **then** ▷ precomputed sum from R_k
    $s \leftarrow R_k(x, y, a)$ ▷ precomputed sum from R_k
  else
    $x_2 \leftarrow (x - \lceil a/2 \rceil) \mod 2^{n+1}$ ▷ circular shift
    $y_2 \leftarrow y + 2^{i-1}$
    $s \leftarrow recSUMk(x, y, \lfloor a/2 \rfloor, i-1, R_k, n, k) + recSUMk(x_2, y_2, \lfloor a/2 \rfloor, i-1, R_k, n, k)$
  return s

We will now introduce a matrix representation for the algorithm $dirPFHT$ with specified values of n, k, L.

4.2.2. Matrix Representation of the 2D Projection Algorithm with FHT Termination and Aggregation

For $i \in \mathbb{Z}_{1,k}$, the element of the tensor $R_i(x, y, a)$ stores the sum value along a pattern of length 2^i along the $0Y$ axis. Here, the pattern initiates at point (x, y), and the pattern's end is shifted by $-a$ along the $0X$ axis. The tensor R_0 is expanded through zero-padding of the original image

$$R_0(x, y, 0) = \begin{cases} I_{2^n}(x, y), & x \in \mathbb{Z}_{0,2^n-1}, y \in \mathbb{Z}_{0,2^n-1}, \\ 0, & x \in \mathbb{Z}_{2^n,2^{n+1}-1}, y \in \mathbb{Z}_{0,2^n-1}. \end{cases} \tag{25}$$

Let us denote the set of indices used in the tensor R_i by $XYA^{(i)}$,

$$(x, y, a) \in XYA^{(i)} = X^{(i)} \times Y^{(i)} \times A^{(i)}, \tag{26}$$

where

$$\begin{aligned} x \in X^{(i)} &= \{j\}_{j=0}^{2^{n+1}-1}, \\ y \in Y^{(i)} &= \{j2^i\}_{j=0}^{2^{n-i}-1}, \\ a \in A^{(i)} &= \{j\}_{j=0}^{2^i-1}. \end{aligned} \tag{27}$$

It is worth noting that

$$\dim(XYA^{(i)}) = 2^{2n+1}, \forall i \in \mathbb{Z}_{0,n}. \tag{28}$$

For all $i \in \mathbb{Z}_{0,n}$, each element of R_i can be represented in terms of $\overline{V}^{(i)}$. More precisely, let $J^{(i)}$ be the bijective function mapping index $(x, y, \alpha) \in XYA^{(i)}$ of the element of tensor R_i to the index j of the corresponding element of vector $\overline{V}^{(i)}$:

$$J^{(i)} : XYA^{(i)} \to \mathbb{Z}_{0,2^{2n+1}-1} \tag{29}$$

is according to formula

$$J^{(i)}(x, y, a) = x + y \cdot 2^{n+1-i} + a \cdot 2^{2n+1-i}. \tag{30}$$

Then, the inverse mapping $H^{(i)} = (J^{(i)})^{-1},\ H^{(i)} : \mathbb{Z}_{0,2^{2n+1}-1} \to XYA^{(i)}$ is computed using

$$\begin{aligned} x &= j \mod 2^{n+1}, \\ y &= (\lfloor \frac{j}{2^{n+1}} \rfloor \mod 2^{n-i}) \cdot 2^i, \\ a &= \lfloor \frac{j}{2^{2n+1-i}} \rfloor. \end{aligned} \tag{31}$$

A $2^{2n+1} \times 2^{2n+1}$ Boolean matrix $\mathbb{B}_i$ allows for the calculation of

$$\overline{V}^{(i)} = \mathbb{B}_i \overline{V}^{(i-1)}. \tag{32}$$

The value of element $\mathbb{B}_i(p,t)$ for $(p,t) \in \mathbb{Z}_{0,2^{2n+1}-1} \times \mathbb{Z}_{0,2^{2n+1}-1}$ is determined by the following formula

$$\mathbb{B}_i(p,t) = \begin{cases} 1, \text{ if} & t = J^{(i-1)}(x_p, y_p, \lfloor a_p/2 \rfloor) \vee \\ & t = J^{(i-1)}((x_p - \lceil a_p/2 \rceil) \mod 2^{n+1}, y_p + 2^{i-1}\lfloor a_p/2 \rfloor), \\ 0, \text{ otherwise,} & \end{cases} \tag{33}$$

where $(x_p, y_p, a_p) = H^{(i)}(p)$.

In each row and each column of the matrix $\mathbb{B}_i$, there are exactly two ones; therefore,

$$C_{\mathbb{1}}(\mathbb{B}_i) = 2H(\mathbb{B}_i) = 2 \cdot 2^{2n+1} = 2^{2n+2}. \tag{34}$$

According to the definition (9), the number of summations for the matrix $\mathbb{B}_i$ is

$$C_{Summ}(\mathbb{B}_i) = C_{\mathbb{1}}(\mathbb{B}_i) - H(\mathbb{B}_i) = 2^{2n+2} - 2^{2n+1} = 2^{2n+1}. \tag{35}$$

For some n, k, the discrete straight line given by (x, a) defines a set $U_n^k(x,a)$ of indices for elements in R_k, the values of which are employed to aggregate sums during the recursive call *recSUMk*:

$$U_n^k(x,a) = \{(x_j, y_j, a_j)\}_{j=0}^{2^{n-k}-1} \in XYA^{(k)}. \tag{36}$$

The aggregating matrix $\mathbb{S}$ is of size $q \times 2^{2n+1}$ and allows for the computation of the vector of sums

$$\overline{s} = \mathbb{S}\overline{V}^{(k)}. \tag{37}$$

The values of the elements in $\mathbb{S}$ are determined by the set of discrete straight lines (L):

$$\mathbb{S}(p,t) = \begin{cases} 1, & (x_p, a_p) \in L \wedge H^{(k)}(t) \in U_n^k(x_t, a_t), \\ 0, & \text{otherwise.} \end{cases} \tag{38}$$

According to the algorithm, each row of matrix $\mathbb{S}$ contains precisely 2^{n-k} ones; hence,

$$C_{\mathbb{1}}(\mathbb{S}) = q2^{n-k} \tag{39}$$

and following the definition in (9), the count of summations for matrix $\mathbb{S}$ is

$$C_{Summ}(\mathbb{S}) = C_{\mathbb{1}}(\mathbb{S}) - H(\mathbb{S}) = q2^{n-k} - q = q(2^{n-k} - 1). \tag{40}$$

The sequential application of (32) and (37) leads to the matrix representation of *dirPFHT*, which takes the form

$$\overline{s} = \mathbb{S}\mathbb{B}_k \ldots \mathbb{B}_1 \overline{V}^{(0)}. \tag{41}$$

The algorithm's computational complexity, considering both (35) and (40), is determined as follows:

$$C_{Summ}(\mathbb{SB}_k \dots \mathbb{B}_1) = C_{Summ}(\mathbb{S}) + \sum_{i=1}^{k} C_{Summ}(\mathbb{B}_i) = q(2^{n-k} - 1) + k2^{2n+1}. \tag{42}$$

4.2.3. Complexity Analysis of 2D Projection Algorithm with FHT Termination and Aggregation

Given that projection directions are uniformly distributed, let the parameter $\alpha \in (0, 1] \in \mathbb{R}$ determine the proportion of directions compared to the maximum possible number (2^n for each type of lines). This results in the number of distinct discrete straight lines for each type, as expressed below

$$q = \dim(\bar{s}) = 2^n \alpha 2^n = \alpha 2^{2n}. \tag{43}$$

Considering the four types of lines in the 2D scenario, the total number of summations, denoted as $T_{\mathcal{B}}(n, k, \alpha)$, in the implementation (42), is as follows

$$\begin{aligned} T_{\mathcal{B}}(n, k, \alpha) &= 4C_{Summ}(\mathbb{SB}_k\mathbb{B}_{k-1} \dots \mathbb{B}_1) = 4\alpha 2^{2n}(2^{n-k} - 1) + 4k2^{2n+1} \\ &= \alpha 2^{3n-k+2} - \alpha 2^{2n+2} + k2^{2n+3} = 2^{2n+2}(\alpha(2^{n-k} - 1) + 2k). \end{aligned} \tag{44}$$

Note that

$$T_{\mathcal{B}}(n, 0, 1) = 2^{2n+2}(2^n - 1) \tag{45}$$

for naive summation along all discrete lines, considering image padding, and for the full FHT in the Brady–Yong method (without aggregation):

$$T_{\mathcal{B}}(n, n, 1) = n2^{2n+3}. \tag{46}$$

Next, let us explore how k and α influence the number of operations, introducing the following notation

$$\delta = n - k, \delta \in \mathbb{Z}_{0,n}. \tag{47}$$

We will analyze the normalized difference $\Delta_n^{\mathcal{B}}(\delta, \alpha)$ between the complexity of the proposed algorithm $T_{\mathcal{B}}(n, n - \delta, \alpha)$ and the original Brady–Yong algorithm $T_{\mathcal{B}}(n, n, 1)$, normalized by factor 2^{2n+2} (representing the number of discrete straight lines for all types and shifts),

$$\Delta_n^{\mathcal{B}}(\delta, \alpha) \stackrel{\text{def}}{=} \frac{1}{2^{2n+2}}(T_{\mathcal{B}}(n, n - \delta, \alpha) - T_{\mathcal{B}}(n, n, 1)) = \alpha(2^{\delta} - 1) - 2\delta. \tag{48}$$

Without aggregation ($\delta = 0$), the complexities coincide: $\Delta_n^{\mathcal{B}}(0, \alpha) = 0$. To analyze the function's properties, let us take the second derivative of $\Delta_n^{\mathcal{B}}(\delta, \alpha)$ with respect to δ. This leads to

$$\frac{\partial \Delta_n^{\mathcal{B}}}{\partial \delta} = \alpha 2^{\delta} \ln 2 - 2, \tag{49}$$

$$\frac{\partial^2 \Delta_n^{\mathcal{B}}}{\partial^2 \delta} = \alpha 2^{\delta} \ln^2 2. \tag{50}$$

With $\frac{\partial^2 \Delta_n^{\mathcal{B}}}{\partial^2 \delta} > 0$, the function $\Delta_n^{\mathcal{B}}(\delta, \alpha)$ is convex with respect to δ. Let us determine the minimum value of the expression with respect to δ, yielding the following

$$\frac{\partial \Delta_n^{\mathcal{B}}}{\partial \delta} = \alpha 2^{\delta} \ln 2 - 2 = 0, \tag{51}$$

$$2^{\delta} = \frac{2}{\alpha \ln 2}, \tag{52}$$

$$\delta = \log_2 \frac{2}{\alpha \ln 2} = 1 - \log_2(\alpha \ln 2). \tag{53}$$

Therefore, the function $\Delta_n^{\mathcal{B}}(\delta, \alpha)$ attains its minimum at

$$\delta_{min}(\alpha) = 1 - \log_2 \ln 2 - \log_2 \alpha. \tag{54}$$

The optimal value, δ_{opt}, can be approximated by rounding δ to the nearest integer:

$$\delta_{opt}(\alpha) \stackrel{\text{def}}{=} \underset{\delta \in Z_{0,n}}{\operatorname{argmin}} \Delta_n^{\mathcal{B}}(\delta, \alpha). \tag{55}$$

The optimal iteration for terminating the summation is

$$k_{opt}(\alpha) = n - \delta_{opt}(\alpha). \tag{56}$$

Table 1 provides the values of δ_{opt} for various α.

Table 1. Dependence of values δ_{opt} on α.

α	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
δ_{opt}	5	4	3	3	3 or 2	2	2	2	2	2 or 1

Figure 4 illustrates the dependency of $\Delta_n^{\mathcal{B}}(\delta, \alpha)$ on δ.

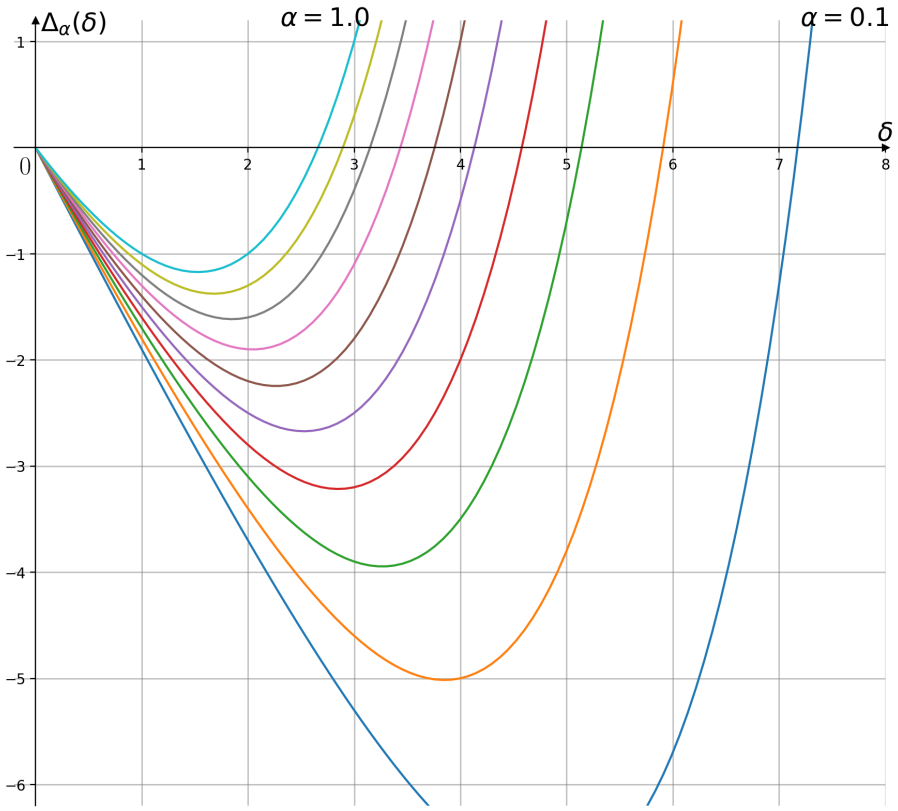

Figure 4. $\Delta_n^{\mathcal{B}}(\delta, \alpha)$ as a continuous function of δ for selected α = 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0. Various colours correspond to different values of parameter α. Values of α decrease from left to right.

Our analysis demonstrates the advantage of the proposed 2D projection algorithm with FHT termination and aggregation compared to the classical Brady–Yong algorithm. Moreover, when α, the proportion of projections used in reconstruction, decreases, the potential computational gain grows.

4.2.4. Transposition of the 2D Projection Algorithm with FHT Termination and Aggregation

To develop a fast algorithm for the calculation of transposed operator, we will analyze the 2D projection algorithm with FHT termination and aggregation (see Section 4.2.1) as a sequential application of partial sum computations for sub-patterns followed by aggregation. In line with the general transposition scheme for summation-based algorithms, the transposition of this algorithm involves executing the transposed stages in reverse order:

(1) "Spreading" the value of each ray-sum along each pattern across all sub-patterns of length 2^k (summing value to all sub-patterns of length 2^k that constitute it);
(2) Transposing the algorithm for computing partial sums.

The implementation of this algorithm, denoted as *revPFHT* (Reversed Partial Fast Hough Transform), computes the transposed operator for the vector of ray-sums $\overline{s} = (s_i)_{i=0}^{q-1}$ and the set of discrete straight lines $L = \{(x_i, a_i)\}_{i=0}^{q-1}$ within the specified range of line directions. Key stages:

1. Preparation of the tensor R_k;
2. Utilizing *revSUMk* (transpose of aggregation algorithm *dirSUMk*);
3. Employing *revPFHT* (transpose of algorithm *dirPFHT* for computing sums over patterns of length 2^k).

Algorithm 5 provides pseudocode for transposition two-dimensional projection Algorithm 1.

Algorithm 5 Transpose of the algorithm *dirPFHT* for calculating sums along patterns of length 2^k

procedure $revPFHT(L, s, n, k)$
  $R_k(x, y) \leftarrow 0 \quad \forall\, x \in \mathbb{Z}_{0,2^{n+1}-1}, y \in \mathbb{Z}_{0,2^n-1}$
  $R_k \leftarrow revSUMk(s_j, x_j, a_j, R_k, n, k)\ \forall j \in \mathbb{Z}_{1,q}$ ▷ transpose of the aggregation algorithm
  $R_0 \leftarrow revPFHTk(R_k, n, k)$ ▷ transpose of the summation algorithm
  $I_{2^n}(x, y) \leftarrow R_0(x, y)\ \forall\, x \in \mathbb{Z}_{0,2^n-1}, y \in \mathbb{Z}_{0,2^n-1}$ ▷ copying the result
  return I_{2^n}

Given n and k, the *revSUMk* Algorithm 6, for a discrete straight line specified by (x, a), stores the value s in the tensor R_k corresponding to the sums over sub-patterns of length 2^k using a recursive call to the function *recSPRk*.

Algorithm 6 Transpose of the aggregation algorithm *dirSUMk*

procedure $revSUMk(s, x, a, R_k, n, k)$
  $R_k \leftarrow recSPRk(s, x, 0, a, n, R_k, n, k)$
  return R_k

The Algorithm 7 *recSPRk* recursively "spreads" (adds) the value s across (to) all elements of tensor R_k, which are used to calculate the sum along the discrete straight line specified by (x, a).

Algorithm 7 Algorithm for recursive spreading for computing the transposed operator

procedure $recSPRk(s, x, y, a, i, R_k, n, k)$
  if $i = k$ **then**
    $R_k(x, y + a) \leftarrow R_k(x, y + a) + s$
  else
    $R_k \leftarrow recSPRk(s, x, y, \lfloor a/2 \rfloor, i - 1, R_k, n, k)$
    $R_k \leftarrow recSPRk(s, x - \lceil a/2 \rceil, y + 2^{i-1}, \lfloor a/2 \rfloor, i - 1, R_k, n, k)$
  return R_k

To transpose the algorithm for calculating partial sums over sub-patterns of length 2^k, we "reverse" the loop with respect to the variable k and replace the sum accumulation with "spreading". The Algorithm 8 *revPFHTk* completes the process of computing the fully transposed operator for the matrix R_k, which contains the results of transposing the aggregation algorithm (the results of calculating sums over sub-patterns of length 2^k).

Algorithm 8 Completing the process of computing the fully transposed operator for the matrix R_k

procedure $revPFHTk(R_k, n, k)$
  for $i = k$ to 0 step -1 **do**
    $R_{i-1}(x, y, 0) \leftarrow 0 \quad \forall\, x \in \mathbb{Z}_{0,2^{n+1}-1}, y \in \mathbb{Z}_{0,2^n-1}$
    for $a = 0$ to $2^i - 1$ **do**
      for $y = 0$ to $2^n - 1$ step 2^i **do**
        for $x = 0$ to $2^{n+1} - 1$ **do**
          $v \leftarrow R_i(x, y, a) + R_{i-1}(x, y, \lfloor a/2 \rfloor)$
          $R_{i-1}(x, y, \lfloor a/2 \rfloor) \leftarrow v$
          $x_2 \leftarrow (x - \lceil a/2 \rceil) \mod 2^{n+1}$ ▷ circular shift
          $v \leftarrow R_i(x, y, a) + R_{i-1}(x_2, y + 2^{i-1}, \lfloor a/2 \rfloor)$
          $R_{i-1}(x_2, y + 2^{i-1}, \lfloor a/2 \rfloor) \leftarrow v$
  return R_0

4.2.5. Analyzing the Complexity of the Transposed 2D Projection Algorithm with FHT Termination and Aggregation

When transposing the 2D projection algorithm with FHT termination and aggregation from (19) and (41), the matrix representation of the transposed operator is as follows

$$\mathbb{B}^T = (\mathbb{S}\mathbb{B}_k \dots \mathbb{B}_1)^T = \mathbb{B}_1^T \dots \mathbb{B}_k^T \mathbb{S}^T. \tag{57}$$

The estimation of the complexity for computing the transposed 2D projection algorithm with FHT termination and aggregation, denoted as $T_{\mathcal{B}^T}(n, k, \alpha)$, is determined (see Section 4.2.3) as

$$T_{\mathcal{B}^T}(n, k, \alpha) = 4C_{Summ}(\mathbb{B}_1^T \dots \mathbb{B}_k^T \mathbb{S}^T) = 4C_{Summ}(\mathbb{B}_1^T \dots \mathbb{B}_k^T) + 4C_{Summ}(\mathbb{S}^T). \tag{58}$$

In general, the matrix $\mathbb{S}$ may have zero columns. Consequently, $\mathbb{S}^T$ may have zero rows. The number of summations in the transposed aggregation algorithm is defined as

$$C_{Summ}(\mathbb{S}^T) = C_{\mathbb{1}}(\mathbb{S}^T) - C_{NZR}(\mathbb{S}^T), \tag{59}$$

where $C_{NZR}(\mathbb{S}^T) > 0$ represents the number of non-zero rows in the matrix $\mathbb{S}^T$.

Let us introduce

$$\hat{T}_{\mathcal{B}^T}(n, k, \alpha) \stackrel{\text{def}}{=} T_{\mathcal{B}^T}(n, k, \alpha) + 4C_{NZR}(\mathbb{S}^T) = 4C_{Summ}(\mathbb{B}_1^T \dots \mathbb{B}_k^T) + 4C_{\mathbb{1}}(\mathbb{S}^T) \tag{60}$$

as an upper bound estimate for the number of summations

$$T_{\mathcal{B}^T}(n, k, \alpha) \leq \hat{T}_{\mathcal{B}^T}(n, k, \alpha). \tag{61}$$

Considering (21) and (35), we can calculate the number of summations in the transposed algorithm for partial sum calculation over sub-patterns:

$$C_{Summ}(\mathbb{B}_1^T \dots \mathbb{B}_k^T) = C_{Summ}(\mathbb{B}_k \dots \mathbb{B}_1) = \sum_{i=1}^{k} C_{Summ}(\mathbb{B}_i) = k2^{2n+1}. \quad (62)$$

By taking into account (39) and (43), we can compute

$$C_{\mathbb{1}}(\mathbb{S}^T) = C_{\mathbb{1}}(\mathbb{S}) = q2^{n-k} = \alpha 2^{2n}2^{n-k} = \alpha 2^{3n-k}. \quad (63)$$

Substituting (62) and (63) into (60), we get

$$\hat{T}_{\mathcal{B}^T}(n,k,\alpha) = k2^{2n+3} + \alpha 2^{3n-k+2} = 2^{2n+2}(2k + \alpha 2^{n-k}). \quad (64)$$

When transposing the original Brady algorithm, aggregation is not performed ($k = n, \alpha = 0$). Thus, the complexity estimate is as follows:

$$T_{\mathcal{B}^T}(n,n,0) = \hat{T}_{\mathcal{B}^T}(n,n,0) = T_{\mathcal{B}}(n,n,0) = 2^{2n+3}n. \quad (65)$$

Similarly to the analysis of the forward operator calculation algorithm (Section 4.2.3) let us examine the normalized difference $\Delta_n^{\mathcal{B}^T}(\delta,\alpha)$ between the numbers of required operations

$$\Delta_n^{\mathcal{B}^T}(\delta,\alpha) \stackrel{\text{def}}{=} \frac{1}{2^{2n+2}}(\hat{T}_{\mathcal{B}^T}(n,n-\delta,\alpha) - T_{\mathcal{B}^T}(n,n,0)) = 2k + \alpha 2^{n-k} - 2n = \alpha 2^{\delta} - 2\delta. \quad (66)$$

Analytically (similarly Section 4.2.3), we can demonstrate that

$$\forall n \in \mathbb{Z}_{2,\infty} \quad \forall 0 < \alpha < 1 \quad \exists \delta \in \mathbb{Z}_{1,n-1} \to \Delta_n^{\mathcal{B}^T}(\delta,\alpha) < 0. \quad (67)$$

Hence, when $\alpha < 1$, the proposed algorithm (*revPFHT*) for computing the transposed 2D projection operator is computationally more efficient when compared to the transposed Brady–Yong algorithm.

Figure 5 illustrates the dependency of $\Delta_n^{\mathcal{B}^T}(\delta,\alpha)$ on δ.

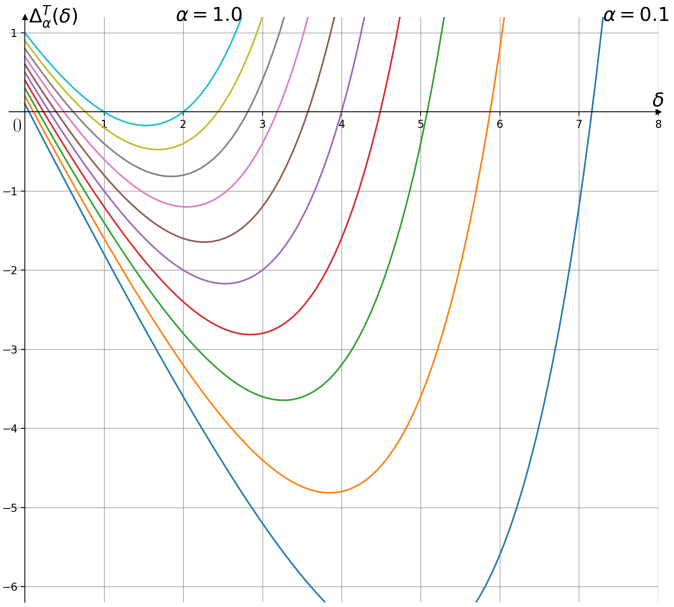

Figure 5. $\Delta_n^{\mathcal{B}^T}(\delta,\alpha)$ as a continuous function of δ for selected values of parameter $\alpha = 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0$. Various colours correspond to different values of parameter α. Values of α decrease from left to right.

4.3. Fundamental 3D FHT Concepts

For detailed definitions of discrete patterns and extensions of the Brady–Yong algorithm for 3D straight lines, please refer to [87]. In 3D, the image is represented by a cubic matrix, where each element, called a voxel, is a unit cube with a side length of 1. The (s, t) parameterization classifies 3D lines into three types based on their alignment with respective coordinate axes, as outlined in [87]. A straight line is deemed predominantly aligned with a specific axis if the acute angle between it and that axis is less than or equal to the angle between this straight line and the other axes. In the following discussion, we focus on lines primarily aligned with the $0Z$ axis. Each of these straight lines is defined by two points, $(s_X, s_Y, 0)$ and $(s_Y + t_X, s_Y + t_Y, 2^n - 1)$, through which it passes (see Figure 6).

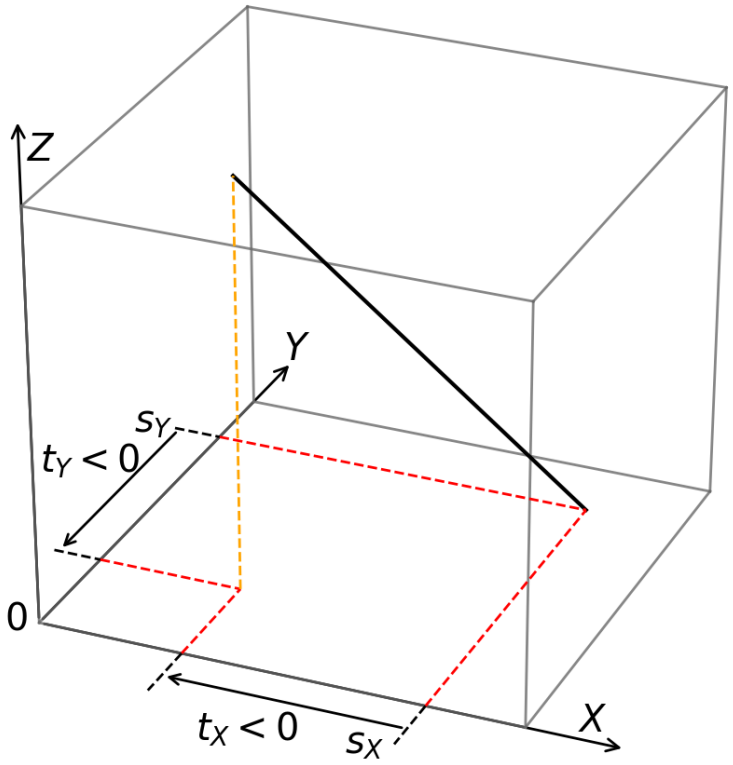

Figure 6. Illustration of parametrization predominantly $0Z$-axis oriented line in 3D, $t_X < 0$, $t_Y < 0$. Different coloured dashed lines demonstrate the values of parameters used in parametrization.

Moreover, lines within each type are further classified into four subtypes based on the signs of parameters t_X and t_Y, determined by their orientation relative to the coordinate axes $0X$ and $0Y$. In this study, we consider cases where there is a negative slope along both axes, i.e., $t_X < 0, t_Y < 0$.

4.4. Fast 3D Forward Projection and Back-Projection Algorithm with FHT and Aggregation

4.4.1. A Fast 3D Forward Projection Operator Algorithm for Arbitrary Set of Straight Lines with Termination at k-th Iteration and Aggregation

In [35,87], a technique based on FHT termination and aggregation was introduced for the efficient computation of the projection operator in 3D, akin to the $dirPFHT$ algorithm detailed in Section 4.2.1. Now, let us discuss the implementation of the $dirPFHT3D$ Algorithm 9, introduced in [35]. Its primary stages involve padding the original image I with zeros, computing partial sums, and aggregating the sum for each straight line from L.

Algorithm 9 Forward projection operator computation in 3D

procedure $dirPFHT3D(L, I, n, k)$
  $R_0(x, y, z, 0, 0) \leftarrow I_{2^n}(x, y, z)\ \forall x \in \mathbb{Z}_{0,2^n-1}, y \in \mathbb{Z}_{0,2^n-1}, z \in \mathbb{Z}_{0,2^n-1}$ ▷ copy the original data
  $R_0(x, y, z, 0, 0) \leftarrow 0\ \forall x \in \mathbb{Z}_{2^n,2^{2n}-1}, y \in \mathbb{Z}_{2^n,2^{2n}-1}, z \in \mathbb{Z}_{0,2^n-1}$ ▷ zero padding
  $R_k \leftarrow dirPFHTk3D(R_0, n, k)$ ▷ compute partial sums (along patterns with length 2^k)
  $s_j \leftarrow dirSUMk3D(x_j, y_j, a_j, b_j, R_k, n, k)\ \forall j \in \mathbb{Z}_{0,q-1}(x_j, y_j, a_j, b_j) \in L$ ▷ aggregation of sums along discrete straight lines from L
  return s

The tensor element $R_i(x, y, z, a, b)$ encapsulates the summation over a pattern of length 2^i. The pattern commences at point (x, y, z), and its terminus shifts along the $0X$ axis by $-a$ and along the $0Y$ axis by $-b$. Tensor R_0 represents the original image padded with zeros. The set of voxel indices employed at the i-th step of the algorithm in tensor R_i,

$$(x, y, z, a, b) \in XYZAB^{(i)} = X^{(i)} \times Y^{(i)} \times Z^{(i)} \times A^{(i)} \times B^{(i)}, \tag{68}$$

is defined as follows

$$\begin{aligned} x \in X^{(i)} &= \{j\}_{j=0}^{2^{n+1}-1} \\ y \in Y^{(i)} &= \{j\}_{j=0}^{2^{n+1}-1} \\ z \in Z^{(i)} &= \{j \cdot 2^i\}_{j=0}^{2^{n-i}-1} \\ a \in A^{(i)} &= \{j\}_{j=0}^{2^i-1} \\ b \in B^{(i)} &= \{j\}_{j=0}^{2^i-1}. \end{aligned} \tag{69}$$

The Algorithm 10 for computing partial sums, denoted as $dirPFHTk3D$, mirrors the previously described 2D version.

Algorithm 10 Partial Sums Computation Algorithm in 3D

procedure $dirPFHTk3D(R_0, n, k)$
  for $i = 1$ to k **do** ▷ along the pattern length 2^i
    for $a = 0$ to $2^i - 1$ **do** ▷ along the absolute $0X$ shift of the pattern end
      for $b = 0$ to $2^i - 1$ **do** ▷ along the absolute $0Y$ shift of the pattern end
        for $x = 0$ to $2^{n+1} - 1$ **do** ▷ along the $0X$ shift of the pattern beginning
          for $y = 0$ to $2^{n+1} - 1$ **do** ▷ along the $0Y$ shift of the pattern beginning
            for $z = 0$ to 2^{n-i} step 2^i **do** ▷ along the $0Z$ shift of the pattern beginning
              $x_2 \leftarrow (x - \lceil \frac{a}{2} \rceil) \mod 2^{n+1}$ ▷ circular shift along $0X$
              $y_2 \leftarrow (y - \lceil \frac{b}{2} \rceil) \mod 2^{n+1}$ ▷ circular shift along $0Y$
              $z_2 \leftarrow z + 2^{i-1}$
              $R_i(x, y, z, a, b) \leftarrow R_{i-1}(x, y, z, \lfloor \frac{a}{2} \rfloor, \lfloor \frac{b}{2} \rfloor) + R_{i-1}(x_2, y_2, z_2, \lfloor \frac{a}{2} \rfloor, \lfloor \frac{b}{2} \rfloor)$
  return R_k

The Algorithm 11 $dirSUMk3D$ operates on the tensor R_k, containing sums over sub-patterns of length 2^k. It conducts aggregation of sums along a discrete 3D straight line with parameters (x, y, a, b), using a recursive call to $recSUMk3D$.

Algorithm 11 Algorithm for 3D sums aggregation

1: **procedure** $dirSUMk3D(x, y, a, b, R_k, n, k)$
2: **return** $recSUMk3D(x, y, 0, a, b, n, R_k, n, k)$

The Algorithm 12 $recSUMk3D$ recursively accumulates sums along the discrete straight line given by (x, y, a, b) from the precomputed partial sums over patterns of length $2^i, i \in \mathbb{Z}_{k,n}$, stored in R_k.

Algorithm 12 Recursive call for sums aggregation over patterns in 3D case

1: **procedure** $recSUMk3D(x, y, z, a, b, i, R_k, n, k)$
2: **if** $i = k$ **then**
3: $s \leftarrow R_k(x, y, z, a, b)$ ▷ precomputed sum from R_k
4: **else**
5: $x_2 \leftarrow (x - \lceil \frac{a}{2} \rceil) \mod 2^{n+1}$ ▷ circular shift along $0X$
6: $y_2 \leftarrow (y - \lceil \frac{b}{2} \rceil) \mod 2^{n+1}$ ▷ circular shift along $0Y$
7: $z_2 \leftarrow z + 2^{i-1}$
8: $s \leftarrow recSUMk3D(x, y, z, \lfloor \frac{a}{2} \rfloor, \lfloor \frac{b}{2} \rfloor, i - 1, R_k, n, k) + recSUMk3D(x_2, y_2, z_2, \lfloor \frac{a}{2} \rfloor, \lfloor \frac{b}{2} \rfloor, i - 1, R_k, n, k)$
9: **return** s

Next, we will establish the matrix representation for the algorithm $dirPFHT3D$ given n, k, L.

4.4.2. Matrix Representation of 3D Forward Projection Operator Computation Algorithm

It is worth noting that

$$\dim(XYZAB^{(i)}) = 2^{3n+2+i}, \forall i \in \mathbb{Z}_{0,n}. \tag{70}$$

For any $i \in \mathbb{Z}_{0,n}$, we can represent R_i as the vector $\overline{V}^{(i)}$. Let the index $(x, y, z, a, b) \in XYZAB^{(i)}$ of the tensor element R_i and the index j of the corresponding element in the vector $\overline{V}^{(i)}$ be bijectively mapped: $J^{(i)} : XYZAB^{(i)} \to \mathbb{Z}_{0,2^{3n+2+i}-1}$. The inverse mapping is denoted as

$$H^{(i)} = (J^{(i)})^{-1}. \tag{71}$$

The Boolean matrix $\mathbb{B}_i$ of size $2^{3n+2+i} \times 2^{3n+1+i}$ allows us to compute

$$\overline{V}^{(i)} = \mathbb{B}_i \overline{V}^{(i-1)}. \tag{72}$$

The value of element $\mathbb{B}_i(p, t)$ for $(p, t) \in \mathbb{Z}_{0,2^{3n+2+i}-1} \times \mathbb{Z}_{0,2^{3n+1+i}-1}$ is determined by the following formula:

$$\mathbb{B}_i(p, t) = \begin{cases} 1, \text{ if } & t = J^{(i-1)}(x_p, y_p, z_p, \lfloor a_p/2 \rfloor, \lfloor b_p/2 \rfloor) \vee \\ & t = J^{(i-1)}((x_p - \lceil a_p/2 \rceil) \mod 2^{n+1}, \\ & (y_p - \lceil b_p/2 \rceil) \mod 2^{n+1}, z_p + 2^{i-1}, \lfloor a_p/2 \rfloor, \lfloor b_p/2 \rfloor), \\ 0, \text{ otherwise,} & \end{cases} \tag{73}$$

where $(x_p, y_p, a_p) = H^{(i)}(p)$. Each row of the matrix $\mathbb{B}_i$ contains exactly 2 ones, hence

$$C_{\mathbb{1}}(\mathbb{B}_i) = 2H(\mathbb{B}_i) = 2 \cdot 2^{3n+2+i} = 2^{3n+3+i}. \tag{74}$$

By substituting into (9), we arrive at

$$C_{Summ}(\mathbb{B}_i) = C_{\mathbb{1}}(\mathbb{B}_i) - H(\mathbb{B}_i) = H(\mathbb{B}_i) = 2^{3n+2+i}. \tag{75}$$

The aggregation matrix $\mathbb{S}$ has dimensions $q \times 2^{3n+2+k}$ and is employed to calculate the vector of sums:

$$\overline{s} = \mathbb{S}\overline{V}^{(k)}. \tag{76}$$

In each row of the matrix $\mathbb{S}$, there are exactly 2^{n-k} ones, thus, based on the (9), we find

$$C_{Summ}(\mathbb{S}) = C_{\mathbb{1}}(\mathbb{S}) - H(\mathbb{S}) = H(\mathbb{S})(2^{n-k} - 1) = q(2^{n-k} - 1). \tag{77}$$

The successive application of (72) and (76) yields the following matrix representation of *dirPFHT3D*

$$\overline{s} = \mathbb{S}\mathbb{B}_k \dots \mathbb{B}_1 \overline{V}^{(0)}. \tag{78}$$

4.4.3. Number of Addition Operations in the 3D Forward Projection Operator Calculation Algorithm with Aggregation

For one type of straight lines, the number of unique discrete straight lines is determined by the range of different origin and end shifts, which in 3D amounts to $N^2N^2 = N^4$.

Let the proportion of considered straight lines be α. Since there are 12 types of straight lines and taking into account (75) and (77), the number of addition operations $T_{\mathcal{B}}(n, k, \alpha)$ when computing the forward operator (Section 4.4.1) can be calculated as follows:

$$\begin{aligned} T_{\mathcal{B}}(n,k,\alpha) =& 12C_{Summ}(\mathbb{S}\mathbb{B}_k\mathbb{B}_{k-1}\dots\mathbb{B}_1) = 12(C_{Summ}(\mathbb{S}) + \sum_{i=1}^{k} C_{Summ}(\mathbb{B}_i)) = \\ & 12(\alpha 2^{4n}(2^{n-k} - 1) + \sum_{i=1}^{k} 2^{3n+2+i}) = 12(\alpha 2^{4n}(2^{n-k} - 1) + 2^{3n+2}\sum_{i=1}^{k} 2^i) = \\ & 3 \cdot 2^{3n+2}(\alpha 2^n(2^{n-k} - 1) + 2^3(2^k - 1)). \end{aligned} \tag{79}$$

Fast summation for all directions without aggregation has a complexity of

$$T_{\mathcal{B}}(n, n, 1) = 3 \cdot 2^{3n+2} \cdot 2^3(2^n - 1). \tag{80}$$

In 3D, forward projection requires summation along $\Theta(N^3)$ straight lines, corresponding to

$$\alpha = \frac{1}{N} = \frac{1}{2^n}, \tag{81}$$

which, when substituted, yields

$$T_{\mathcal{B}}(n, k, \frac{1}{2^n}) = 3 \cdot 2^{3n+2}((2^{n-k} - 1) + 2^3 \cdot (2^k - 1)). \tag{82}$$

Naive summation corresponds to $k = 0$, and the total number of summations during the operator calculation is

$$T_{\mathcal{B}}(n, 0, \frac{1}{2^n}) = 3 \cdot 2^{3n+2}(2^n - 1) = 3 \cdot 2^{3n+2}(2^n - 1). \tag{83}$$

In [35], a theoretical estimate of the asymptotically optimal $k = \frac{n}{2}$ is provided for the summation algorithm along $\Theta(2^{3n})$ lines, substituting which, we get

$$\begin{aligned} T_{\mathcal{B}}(n, \frac{n}{2}, \frac{1}{2^n}) =& 3 \cdot 2^{3n+2}((2^{n-\frac{n}{2}} - 1) + 2^3 \cdot (2^{\frac{n}{2}} - 1)) \\ =& 3^3 2^{3n+2}(2^{\frac{n}{2}} - 1) = \Theta(2^{\frac{7}{2}n}). \end{aligned} \tag{84}$$

The complexity estimate derived from the analysis of the matrix decomposition of the algorithm proposed in [35], $\Theta(2^{\frac{7}{2}n})$, aligns with the theoretical estimate provided by authors [35].

Next, we will construct the algorithm for calculating the transposed operator, achieved by transposing the algorithm introduced in [35] for computing the forward operator in 3D (FHT with aggregation).

4.4.4. Transposing 3D Forward Projection Algorithm with FHT Termination at k-th Iteration and Aggregation

The transposition of the algorithm for computing the forward projection operator in 3D with aggregation closely follows the process in the 2D case (see Section 4.2.4). It entails the sequential application of the transposed aggregation algorithm, referred to as *revSUMk3D*, and the transposed algorithm for computing sums over patterns of length 2^k, denoted as *revPFHT3D*. Although line parameterization becomes more complex, and the aggregating tensor's dimensionality increases, the fundamental structure of the algorithm remains unchanged.

Implemented as a function called *revPFHT3D* Algorithm 13 (Reversed Partial Fast Hough Transform 3D), the algorithm computes the transposed operator. Given parameters n, k, it takes a vector of ray sums $\bar{s} = (s_i)_{i=0}^{q-1}$ for q directions defined as a set of discrete straight lines $L = \{(x_i, y_i, a_i, b_i)\}_{i=0}^{q-1}$, and returns a $2^n \times 2^n \times 2^n$ image I.

Algorithm 13 Algorithm for computing the transposed operator in 3D

1: **procedure** $revPFHT3D(L, s, n, k)$
2: $R_k(x, y, z, a, b) \leftarrow 0 \; \forall x, y \in \mathbb{Z}_{0,2^{n+1}-1}, \; \forall z = j2^k, j \in \mathbb{Z}_{0,2^{n-k}-1}, \; \forall a, b \in \mathbb{Z}_{0,2^k-1}$
3: $R_k \leftarrow revSUMk3D(s_j, x_j, y_j, a_j, b_j, R_k, n, k) \; \forall j \in \mathbb{Z}_{1,q}$ ▷ transposed algorithm for aggregation in 3D
4: $R_0 \leftarrow revPFHTk3D(R_k, n, k)$ ▷ transposed algorithm for summation in 3D
5: $I(x, y, z) \leftarrow R_0(x, y, z, 0, 0) \; \forall x \in \mathbb{Z}_{0,2^n-1}, \; y \in \mathbb{Z}_{0,2^n-1}, \; z \in \mathbb{Z}_{0,2^n-1}$ ▷ copying the result
6: **return** I

The Algorithm 14 *revSUMk3D* is the transposed aggregation algorithm. Given parameters n and k, it adds value s to the elements of tensor R_k, corresponding to partial sums over sub-patterns of length 2^k, along the discrete straight line defined by starting point (x, y) and shift of the end (a, b).

Algorithm 14 Transposed algorithm for aggregation when computing the transposed operator in 3D

1: **procedure** $revSUMk3D(s, x, y, a, b, R_k, n, k)$
2: $R_k \leftarrow recSPRk3D(s, x, y, 0, a, b, n, R_k, n, k)$
3: **return** R_k

The accumulation of partial summation results is accomplished through a recursive call to the Algorithm 15 *recSPRk3D*.

Algorithm 15 Algorithm for recursive call when accumulating partial summation results

procedure $recSPRk3D(s, x, y, z, a, b, i, R_k, n, k)$
  if $i = k$ **then**
    $R_k(x, y, z, a, b) \leftarrow R_k(x, y, z, a, b) + s$
  else
    $x_2 \leftarrow (x - \lceil \frac{a}{2} \rceil) \mod 2^{n+1}$
    $y_2 \leftarrow (y - \lceil \frac{b}{2} \rceil) \mod 2^{n+1}$
    $R_k \leftarrow recSPRk3D(s, x, y, z, \lfloor \frac{a}{2} \rfloor, \lfloor \frac{b}{2} \rfloor, i-1, R_k, n, k)$
    $R_k \leftarrow recSPRk3D(s, x_2, y_2, z + 2^{i-1}, \lfloor \frac{a}{2} \rfloor, \lfloor \frac{b}{2} \rfloor, i-1, R_k, n, k)$
  return R_k

The Algorithm 16 *revPFHTk3D* implements the transposed algorithm for computing partial sums along pattern of size 2^k.

Algorithm 16 Transposed algorithm for computing partial sums along pattern of size 2^k

procedure $revPFHTk3D(R_k, n, k)$
  for $i = k$ to 1 step -1 **do**
    $R_{i-1}(x, y, z, a, b) \leftarrow 0 \; \forall x, y \in \mathbb{Z}_{0,2^{n+1}-1}, \; \forall z = j2^i, j \in \mathbb{Z}_{0,2^{n-i}-1}, \; \forall a, b \in \mathbb{Z}_{0,2^i}$
    for $a = 0$ to $2^i - 1$ **do**
      for $b = 0$ to $2^i - 1$ **do**
        for $x = 0$ to $2^{n+1} - 1$ **do**
          for $y = 0$ to $2^{n+1} - 1$ **do**
            for $z = 0$ to 2^{n-i} step 2^i **do**
              $v \leftarrow R_i(x, y, z, a, b) + R_{i-1}(x, y, z, \lfloor \frac{a}{2} \rfloor, \lfloor \frac{b}{2} \rfloor)$
              $R_{i-1}(x, y, z, \lfloor \frac{a}{2} \rfloor, \lfloor \frac{b}{2} \rfloor) \leftarrow v$
              $x_2 \leftarrow (x - \lceil \frac{a}{2} \rceil) \mod 2^{n+1}$
              $y_2 \leftarrow (y - \lceil \frac{b}{2} \rceil) \mod 2^{n+1}$
              $v \leftarrow R_i(x, y, z, a, b) + R_{i-1}(x_2, y_2, z + 2^{i-1}, \lfloor \frac{a}{2} \rfloor, \lfloor \frac{b}{2} \rfloor)$
              $R_{i-1}(x_2, y_2, z + 2^{i-1}, \lfloor \frac{a}{2} \rfloor, \lfloor \frac{b}{2} \rfloor) \leftarrow v$
  return R_0

Next, let us evaluate the algorithmic complexity of the proposed approach.

4.4.5. Number of Addition Operations in the 3D Transposed Operator Algorithm with Aggregation

Just as in the 2D case (Section 4.2.5) considering (59), the complexity estimate $T_{\mathcal{B}^T}$ for the algorithm *revPFHT3D* computing the transposed operator in 3D can be expressed as

$$\begin{aligned} T_{\mathcal{B}^T}(n,k,\alpha) =& 12 C_{Summ}(\mathbb{B}_1^T \dots \mathbb{B}_k^T \mathbb{S}^T) \\ =& 12 (\sum_{i=1}^{k} C_{Summ}(\mathbb{B}_i^T) + C_{Summ}(\mathbb{S}^T)) \\ =& 12 (\sum_{i=1}^{k} C_{Summ}(\mathbb{B}_i^T) + C_{\mathbb{1}}(\mathbb{S}^T) - C_{NZR}(\mathbb{S}^T)), \end{aligned} \tag{85}$$

where $C_{NZR}(\mathbb{S}^T) > 0$ denotes the count of non-zero rows in the matrix $\mathbb{S}^T$.

Taking into account the matrix structure with the substitution (74), we derive

$$\sum_{i=1}^{k} C_{Summ}(\mathbb{B}_i^T) = \sum_{i=1}^{k} (C_{\mathbb{1}}(\mathbb{B}_i^T) - H(\mathbb{B}_i^T)) = \sum_{i=1}^{k} (2^{3n+i+3} - 2^{3n+i+1}) = 2^{3n+1} 6 (2^k - 1). \tag{86}$$

The transposed algorithm for fast summation across all directions, without aggregation, bears a complexity of

$$T_{\mathcal{B}^T}(n,n,1) = 12\sum_{i=1}^{n} C_{Summ}(\mathbb{B}_i^T) = 12\cdot 2^{3n+1}6(2^n-1) = 9\cdot 2^{3n+4}(2^n-1). \tag{87}$$

To provide an upper-bound estimate $\hat{T}_{\mathcal{B}^T}$ for the complexity of computing the transposed projection operator

$$T_{\mathcal{B}^T} \leq \hat{T}_{\mathcal{B}^T}, \tag{88}$$

let us define

$$\hat{T}_{\mathcal{B}^T} \stackrel{\text{def}}{=} T_{\mathcal{B}^T} + 12C_{NZR}(\mathbb{S}^T) = 12(\sum_{i=1}^{k} C_{Summ}(\mathbb{B}_i^T) + C_{\mathbb{1}}(\mathbb{S}^T)). \tag{89}$$

By substituting (75), (77), and (86), we find

$$\begin{aligned}\hat{T}_{\mathcal{B}^T}(n,k,\alpha) =& 12(2^{3n+1}6(2^k-1)+\alpha 2^{4n}2^{n-k}) = \\ & 3\cdot 2^{3n+2}(12(2^k-1)+\alpha 2^{2n-k}).\end{aligned} \tag{90}$$

In 3D, CT reconstruction methods necessitate the calculation of the operator for $\Theta(N^3)$ straight lines, corresponding to

$$\alpha = \frac{1}{N} = \frac{1}{2^n}. \tag{91}$$

Substituting, we obtain

$$\hat{T}_{\mathcal{B}^T}(n,k,\frac{1}{2^n}) = 3\cdot 2^{3n+2}(12(2^k-1)+2^{n-k}). \tag{92}$$

The transposed algorithm [35] does not implement aggregation, which means $k=0$, so the number of summations is

$$\hat{T}_{\mathcal{B}^T}(n,0,\frac{1}{2^n}) = 3\cdot 2^{3n+2}2^n = 3\cdot 2^{4n+2}. \tag{93}$$

When employing aggregation (similar to the estimation for the forward operator, see Section 4.4.3), we adopt the asymptotically optimal value of $k_{opt} = \frac{n}{2}$, and by substituting it, we acquire

$$\hat{T}_{\mathcal{B}^T}(n,\frac{n}{2},\frac{1}{2^n}) = 3\cdot 2^{3n+2}(12(2^{\frac{n}{2}}-1)+2^{\frac{n}{2}}) = 3\cdot 2^{3n+2}(13\cdot 2^{\frac{n}{2}}-12). \tag{94}$$

Hence, the proposed and analyzed algorithm for fast computation of the transposed projection operator in 3D, obtained by transposing the algorithm proposed in [35], in the practical case of summing over $\Theta(N^3)$ straight lines, exhibits an asymptotic complexity no less favorable than the algorithm outlined in [35] for fast computation of the forward operator $\Theta(N^3\sqrt{N})$ (with N denoting the linear size of the reconstruction).

5. Experiments and Discussion

We tested the proposed general transposition method for summation-based algorithms by implementing efficient forward projection and backprojection operators for the two-dimensional scenario. The implementation, conducted in C++, followed the computational framework outlined in Section 4.2: FHT computation with termination and aggregation. The optimized implementation of the algorithms prompted numerical experiments to gauge the speed and quality of their performance. The realization of the method was performed using our closed libraries for fast image processing to achieve a realistic execution time together with adequate comparison to classical methods. For this reason, unfortunately, the

implementation can not be provided in open access. Although we hope that our detailed pseudo code implementations would be enough to reproduce the results.

We compared the quality of reconstruction in the case of parallel-beam circular CT. As reference methods for CT reconstruction, we employed the classical FBP method [63] and the accelerated FBP for computed tomography image reconstruction (referred to as HFBP) from [86]. The modified HFBP method, utilizing fast forward and backprojection operators obtained through the proposed transposition method, is denoted as HFBP-L. The reconstructions were acquired using the Smart Tomo Engine v.2.0.0 software [90].

For our experiments, we selected the Shepp–Logan phantom model (see Figure 7) with dimensions of $N \times N, N = 1024$. The set of projections simulated measurements with a detector size of N for $4N - 3$ source rotation angles around the object's center, with a uniform angle step from 0 to π. The projections were simulated in parallel geometry of the experiment as a simple forward projection, which can be reproduced by any open-source software for CT simulation.

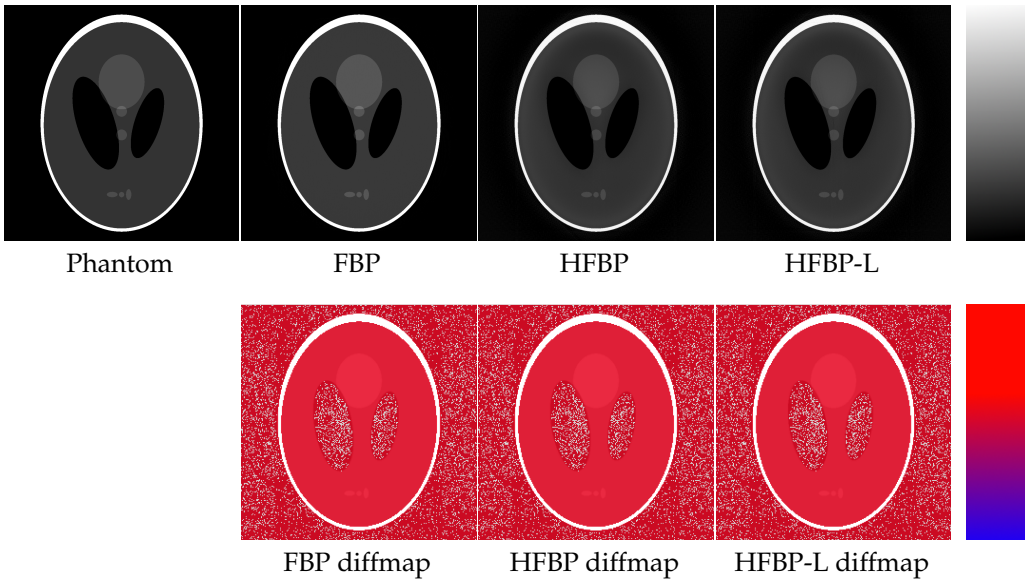

Figure 7. The Shepp–Logan phantom with $N = 1024$, the reconstruction results using the considered methods along with the difference maps compared to the ideal phantom image. The entire set of projection data was utilized for the reconstruction (α = 1.0). The HFBP-L method employed an optimal aggregation depth δ, determined by the formulas for δ_{opt} from Sections 4.2.3 and 4.2.5 for both forward projection and backprojection operators.

The reconstruction results, obtained using the listed methods, are displayed in Figure 7. The entire set of projection data was utilized for the reconstruction. It is important to note that in the case of the HFBP-L method, we applied optimal aggregation depth determined by corresponding formulas for δ_{opt} outlined in Sections 4.2.3 and 4.2.5 for both forward projection and backprojection operators (with $\alpha = 1.0$).

To evaluate the performance of the reconstruction methods (refer to Table 2), we employed numerical metrics including NRMSE, SSIM [91], and STRESS [92]. The NRMSE metric for the reconstructed image $(\hat{r}_i : i \in \mathbb{Z}_{1,N})^T$ is calculated as $\frac{\sqrt{\sum_{i=1}^{N} |\hat{r}_i - r_i|^2}}{\sqrt{\sum_{i=1}^{N} |r_i|^2}}$, where $(r_i : i \in \mathbb{Z}_{1,N})^T$ represents the vector of pixel values in the ideal phantom image.

The consistency in accuracy metrics between the HFBP and HFBP-L reconstruction methods is not coincidental. This confirms the validity of our algorithm implementation and the proposed aggregation approach. The algorithm's accuracy should not be influenced by the aggregation level δ, and the reconstruction outcome should be indistinguishable from the result of the same algorithm without aggregation, as evidenced by matching accuracy metrics. At the same time, according to the performance measurements, both

HFBP and HFBP-L fall slightly behind the classical FBP algorithm, as they employ a less precise straight line approximation for the sake of speed. Nonetheless, the visual quality of reconstruction using HFBP and HFBP-L is quite satisfactory, with no significant distortions observed in the images (refer to Figure 7).

Table 2. Reconstruction accuracy assessment for $N = 1024$. NRMSE: Smaller values indicate higher accuracy; SSIM $\in [0, 1]$: larger values indicate higher accuracy; STRESS: smaller values indicate higher accuracy. In the HFBP-L method, the aggregation depth is optimized and determined by formulas similar to Equation (55).

Metrics	$\mathbf{FBP}_{\alpha=1.0}$	$\mathbf{HFBP}_{\alpha=1.0}$	$\mathbf{HFBP\text{-}L}_{\alpha=1.0}$	$\mathbf{FBP}_{\alpha=0.1}$	$\mathbf{HFBP}_{\alpha=0.1}$	$\mathbf{HFBP\text{-}L}_{\alpha=0.1}$
NRMSE	0.16	0.16	0.16	0.19	0.25	0.25
SSIM	0.93	0.77	0.77	0.56	0.33	0.33
STRESS	0.07	0.16	0.16	0.11	0.24	0.24

Comparing algorithm speed, it is crucial to highlight that the distinction between HFBP and HFBP-L methods lies solely in the computation of the backprojection operator. HFBP employs the transposed Brady–Yong algorithm without aggregation, whereas HFBP-L utilizes the *revPFHT* algorithm (refer to Section 4.2.5). When discussing execution time, the number of operations, or the algorithms' asymptotic complexity, we specifically focus on the projection operators, excluding additional consistent costs for both methods.

Both HFBP and HFBP-L share identical asymptotic algorithmic complexities. However, HFBP-L displays superior speed, characterized by a constant factor. For instance, at $\alpha = 1.0$, HFBP-L requires at least $0.1N^2$ fewer addition operations than HFBP. And at $\alpha = 0.1$, this difference is at least $22.5N^2$ (assuming optimal aggregation depth as per formula (55)).

Experimental measurements of algorithm execution time were conducted on a Windows 10 system with x64 architecture, driven by an AMD Ryzen 7 2700X processor clocked at 3.70 GHz, featuring a 16 MB L3 cache, and 64 GB of RAM. All measurements were executed in a single thread, and each measurement was repeated 1000 times. The recorded time values were averaged with an assessment of the standard deviation. The recorded time corresponds to the computation duration of the projection operation. Additionally, although computing the forward projection operator is not essential for HFBP and HFBP-L algorithms, corresponding values were measured for completeness and for comparison with theoretical dependencies. These values are also annotated in reference to their respective algorithm.

The computation time of both the forward projection and backprojection operators for the HFBP method is independent of α and can be represented by a single value. Specifically, the computation time for the forward projection operator is $t_{dir}^{HFBP} = 44.0 \pm 1.1$ ms, while for the backprojection operator, it is $t_{rev}^{HFBP} = 79.8 \pm 1.8$ ms.

For the HFBP-L method, the computation time is influenced by both the number of projections, denoted by the parameter α, and the aggregation depth δ. The computation time for the forward projection and backprojection operators is detailed in Tables 3 and 4 respectively. Case $\delta = 0$ corresponds to computing the operator fully using the Brady–Yong method without termination.

As the number of projection directions increases, the overall computation time for the forward projection operator also rises. In the range of $0 < \alpha \leq 0.3$, aggregation leads to time savings (refer to Table 3). The most significant reduction in computation time for the forward projection operator, by 15.8%, is achieved at $\alpha = 0.01, \delta = 3$.

In the computation of the backprojection operator within the HFBP-L method, optimizing the aggregation depth speeds up the process for α values within the range of $0 < \alpha \leq 0.4$ for the considered phantom (see Table 4). The highest acceleration, 21.5%, is attainable at $\alpha = 0.01, \delta = 3$ for computing the backprojection operator.

It is worth noting that even with a zero aggregation level ($\delta = 0$), the computation time of the operators is influenced by the number of projections, despite the absence of aggregation. This arises from additional overhead costs related to the explicit formation of the straight line list.

Table 3. The average time t_{dir}^{HFBP-L} of computing the forward projection operator for a phantom with $N = 1024$. Here, α signifies the proportion of the total $(4N - 3)$ projections. Time is measured in milliseconds. The minimum computation time for the forward projection operator, which corresponds to the experimental optimal aggregation depth, is emphasized in bold.

	$\delta = 0$	$\delta = 1$	$\delta = 2$	$\delta = 3$	$\delta = 4$	$\delta = 5$
$\alpha = 0.01$	42.66 ± 1.9	38.24 ± 1.9	36.90 ± 1.8	$\mathbf{35.92 \pm 1.8}$	36.27 ± 1.8	38.29 ± 1.7
$\alpha = 0.02$	43.07 ± 2.0	39.03 ± 1.9	37.66 ± 1.8	$\mathbf{37.33 \pm 1.8}$	38.91 ± 1.7	43.43 ± 1.7
$\alpha = 0.03$	42.91 ± 1.9	38.97 ± 1.8	$\mathbf{38.13 \pm 1.8}$	38.22 ± 1.7	41.17 ± 1.7	48.99 ± 1.7
$\alpha = 0.05$	43.87 ± 2.0	39.71 ± 2.0	$\mathbf{39.64 \pm 1.8}$	40.85 ± 1.7	45.72 ± 1.7	59.82 ± 1.7
$\alpha = 0.1$	44.56 ± 1.9	$\mathbf{41.80 \pm 1.9}$	42.85 ± 1.8	46.93 ± 1.8	57.74 ± 1.7	86.49 ± 1.8
$\alpha = 0.2$	46.66 ± 1.9	$\mathbf{45.17 \pm 1.9}$	49.10 ± 1.7	58.77 ± 1.7	82.56 ± 1.3	139.10 ± 1.5
$\alpha = 0.3$	48.88 ± 1.9	$\mathbf{48.75 \pm 1.8}$	55.32 ± 1.7	70.68 ± 1.3	105.61 ± 1.4	184.75 ± 1.6
$\alpha = 0.4$	$\mathbf{50.55 \pm 1.9}$	51.64 ± 1.8	61.73 ± 1.8	82.56 ± 1.3	130.43 ± 1.4	243.39 ± 1.5
$\alpha = 0.5$	$\mathbf{52.21 \pm 1.9}$	54.97 ± 1.7	66.88 ± 1.6	93.94 ± 1.4	152.30 ± 1.3	296.08 ± 1.6
$\alpha = 0.6$	$\mathbf{54.68 \pm 1.8}$	58.22 ± 1.7	74.69 ± 1.6	106.06 ± 1.4	176.91 ± 1.3	344.08 ± 1.4
$\alpha = 0.7$	$\mathbf{55.98 \pm 1.8}$	61.04 ± 1.8	79.14 ± 1.5	117.01 ± 1.3	200.84 ± 1.1	399.64 ± 1.5
$\alpha = 0.8$	$\mathbf{58.02 \pm 1.9}$	65.02 ± 1.7	85.91 ± 1.5	128.92 ± 1.4	225.00 ± 1.3	446.75 ± 1.5
$\alpha = 0.9$	$\mathbf{60.51 \pm 1.8}$	68.22 ± 1.7	92.12 ± 1.4	141.92 ± 1.3	247.06 ± 1.4	498.56 ± 1.4
$\alpha = 1.0$	$\mathbf{62.45 \pm 1.8}$	70.93 ± 1.5	98.55 ± 1.5	152.94 ± 1.4	268.12 ± 1.4	536.28 ± 2.2

Table 4. The average computation time t_{rev}^{HFBP-L} for computing the backprojection operator during reconstruction using the HFBP-L algorithm for a phantom with $N = 1024$. The parameter α corresponds to a fraction of the total $(4N - 3)$ number of projections. Time is measured in milliseconds. The minimum computation time for the backprojection operator, corresponding to the experimental optimal aggregation depth, is shown in bold.

	$\delta = 0$	$\delta = 1$	$\delta = 2$	$\delta = 3$	$\delta = 4$	$\delta = 5$
$\alpha = 0.01$	64.43 ± 2.2	57.69 ± 2.1	53.86 ± 2.1	$\mathbf{50.58 \pm 2.0}$	51.82 ± 2.1	50.95 ± 2.0
$\alpha = 0.02$	64.85 ± 2.2	58.87 ± 2.1	55.24 ± 2.1	$\mathbf{52.48 \pm 2.1}$	55.25 ± 2.19	58.81 ± 2.1
$\alpha = 0.03$	65.38 ± 2.2	58.80 ± 2.1	55.60 ± 2.0	$\mathbf{53.80 \pm 2.0}$	57.94 ± 2.0	65.90 ± 1.9
$\alpha = 0.05$	65.81 ± 2.3	59.83 ± 2.2	57.11 ± 2.1	$\mathbf{56.92 \pm 2.0}$	65.30 ± 1.9	81.89 ± 1.8
$\alpha = 0.1$	66.82 ± 2.2	61.47 ± 2.1	$\mathbf{61.28 \pm 2.1}$	65.17 ± 1.9	83.76 ± 1.9	120.80 ± 1.8
$\alpha = 0.2$	69.33 ± 2.2	$\mathbf{66.10 \pm 2.1}$	68.76 ± 2.0	83.08 ± 1.9	122.06 ± 2.0	198.38 ± 2.0
$\alpha = 0.3$	72.35 ± 2.1	$\mathbf{70.51 \pm 2.0}$	77.19 ± 2.0	101.70 ± 1.9	158.65 ± 1.9	274.02 ± 2.3
$\alpha = 0.4$	75.02 ± 2.1	$\mathbf{74.34 \pm 2.0}$	86.06 ± 2.0	120.17 ± 2.0	196.88 ± 2.1	349.87 ± 2.5
$\alpha = 0.5$	$\mathbf{77.70 \pm 2.1}$	79.16 ± 2.0	94.90 ± 2.1	138.64 ± 2.0	232.26 ± 2.1	430.20 ± 2.4
$\alpha = 0.6$	$\mathbf{81.57 \pm 2.1}$	83.62 ± 2.0	105.85 ± 2.3	160.70 ± 2.2	271.90 ± 2.2	505.73 ± 2.4
$\alpha = 0.7$	$\mathbf{84.13 \pm 2.1}$	88.03 ± 2.1	114.66 ± 2.1	176.09 ± 2.2	309.91 ± 2.3	587.20 ± 2.5
$\alpha = 0.8$	$\mathbf{87.12 \pm 2.1}$	93.55 ± 2.1	124.95 ± 2.2	194.32 ± 2.4	346.99 ± 2.6	658.04 ± 2.7
$\alpha = 0.9$	$\mathbf{91.21 \pm 2.2}$	99.01 ± 2.2	135.23 ± 2.2	214.12 ± 2.4	382.09 ± 2.7	733.19 ± 2.9
$\alpha = 1.0$	$\mathbf{94.37 \pm 2.2}$	104.53 ± 2.2	145.02 ± 2.2	232.94 ± 2.5	424.16 ± 2.8	811.04 ± 3.1

The time values for computing the forward projection and backprojection operators, provided in Tables 3 and 4, are represented as plots in Figure 8a,b. The plot illustrates the difference in time compared to $\delta = 0$ for the corresponding α value. This facilitates comparison with the theoretical plots in Section 4.2. The key distinction from theoretical calculations lies in the fact that the optimal aggregation depth, δ_{opt}, deviates from the theoretical value (refer to Section 4.2). Moreover, for $\alpha \geq 0.5$, there is no optimal integer aggregation level that would lead to a time improvement in computing the forward projection and backprojection operators. Essentially, in these scenarios, $\delta = 0$ proves to be optimal.

In summary, the conducted experiments validate the theoretically grounded potential for accelerating reconstruction methods. This is achieved through a fast algorithm for computing the forward projection operator using FHT with termination and aggregation, and a fast algorithm for computing the backprojection operator, achieved by applying a common transposition technique for summation-based algorithms to the forward projection calculation. Time is saved by reducing the number of projection frames used in the reconstruction process. The proposed methods enable accelerations of up to 15% for the forward projection operator and 21% for the backprojection operator when compared to fast calculation methods based on the Brady–Yong algorithm. This acceleration is most pronounced when dealing with a small number of angles and utilizing the optimal aggregation depth $\delta = \delta_{opt}$. This outcome holds particular significance in the realm of few-view CT.

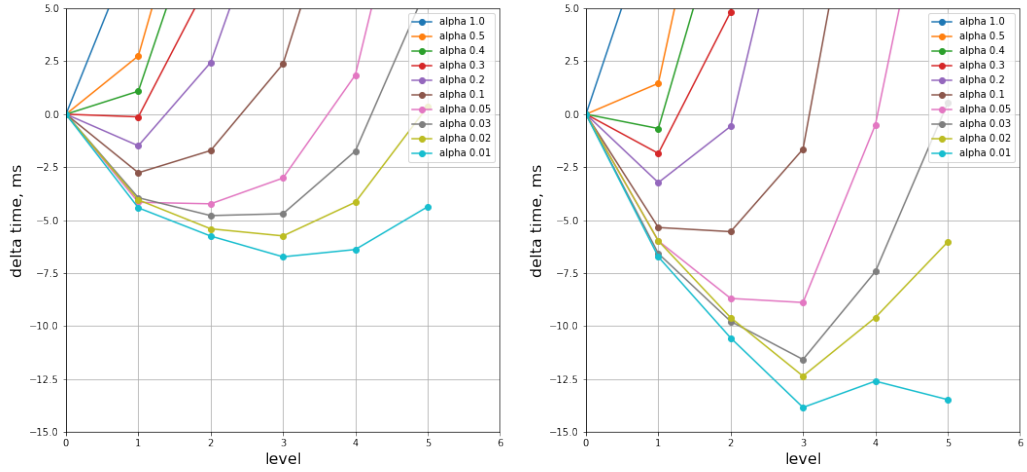

Figure 8. The computation time for the forward projection and backprojection operators with FHT and aggregation at a specified depth *level* (the HFBP-L method) as the difference from the time at $\delta = 0$ for the corresponding α value. (**a**) Dependency for the forward projection operator, (**b**) Dependency for the backprojection operator.

6. Conclusions

This work introduces a versatile method for transposing summation-based algorithms, which rely exclusively on addition operations. This method facilitates the efficient computation of the transpose of linear operators represented as Boolean matrices, assuming a known fast computation algorithm is available. Importantly, this transposition technique maintains the asymptotic complexity of the algorithms, ensuring consistent asymptotic algorithmic complexity for both the original and derived algorithms.

The summation-based transposition method holds significant promise in computer tomography, particularly in algorithms for computing forward projection and backprojection operators, where matrices are transposed in pairs. By applying this method to a known algorithm for computing the forward projection operator, we can derive an algorithm for computing the backprojection operator. The number of addition operations in the resulting algorithm align with the copy operations in the original, and vice versa.

In this study, for the first time, we present fast summation-based algorithms for computing forward projection and backprojection operators in 2D tomographic reconstruction, tailored for parallel-beam CT, especially suitable for few-view CT (FVCT). Although the asymptotic complexity remains at $\Theta\left(n^2 \log n\right)$ (where n represents the linear size of the reconstruction), consistent with the Brady–Yong algorithm, our proposed algorithms are superior in terms of constant factors.

We also extend the method to cone-beam 3D CT. Previously, a forward projection algorithm with a complexity of $\Theta\left(n^{7/2}\right)$ was introduced for this setup [35]. While the number and linear dimensions of the projections scale proportionally with n, a fast algorithm for backprojection was lacking. In this study, we devised such an algorithm, also with a complexity of $\Theta\left(n^{7/2}\right)$.

Therefore, the summation-based transposition method serves as a versatile approach for developing fast algorithms in CT reconstruction, applicable across various measurement schemes, both in two and three-dimensional scenarios. Implementing this method with existing forward projection algorithms, such as [35], adaptable to diverse measurement schemes, facilitates the creation of fast algorithms for computing the backprojection operator in any measurement setup.

We implemented the computation of the backprojection operator by transposing the algorithm for computing the forward projection operator based on FHT with termination and aggregation. This allowed us to assess the accuracy and speed of the new algorithm, comparing it with theoretical dependencies and experimental baselines. The results confirm the theoretically justified potential for accelerating classical reconstruction methods using this transposition method. They also suggest a minimum time depending on the chosen aggregation level. Notably, applying the transposed algorithm, which employs FHT with termination and aggregation, to a fraction of 0.01 of the considered projection directions during reconstruction led to a 15% increase in speed for computing the forward projection operator and a 21% increase for the backprojection operator, with visually insignificant degradation of the result.

Further research into the dynamics of the parameters, particularly the fraction of projections used for reconstruction, where the application of aggregation allows for time savings, is considered worthwhile. This should be explored for various linear sizes of reconstruction. Moreover, the intriguing potential of applying the proposed summation-based transposition method to algorithms computing the forward projection operator, particularly those based on cutting-edge, accurate, fast algorithms for Hough transforms, warrants attention. Employing this method in tandem with precise, fast algorithms such as those in [55] could offer a promising avenue for achieving both speed and accuracy in reconstruction simultaneously.

Author Contributions: Conceptualization, D.N. and D.P.; methodology, D.N., D.P., M.G., P.K. and D.K.; software, P.K. and M.G.; validation, D.K., A.I., M.G. and P.K.; formal analysis, D.K., M.C. and P.K.; investigation, P.K., M.C. and D.P.; resources, A.I. and M.G.; data curation, D.P. and A.I.; writing—original draft preparation, D.N., M.G., D.K., M.C., A.I., P.K. and D.P.; writing—review and editing, D.N., M.G., D.K. and M.C.; visualization, P.K., D.P., A.I. and M.G.; supervision, D.N., M.C. and D.P.; project administration, D.N., M.C. and D.P.; funding acquisition, M.C. All authors have read and agreed to the published version of the manuscript.

Funding: The work was supported by the Russian Science Foundation, project no. 23-21-00524.

Data Availability Statement: Data are contained within the article.

Conflicts of Interest: Authors Dmitry Polevoy, Marat Gilmanov, Danil Kazimirov, Marina Chukalina, Anastasia Ingacheva, Petr Kulagin, Dmitry Nikolaev were partly employed by the company Smart Engines Service LLC. The research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

CT	computed tomography
FVCT	few-view computed tomography

Nomenclature

$\lfloor \cdot \rfloor$	rounding down to the nearest integer (integer part or floor function)
$\lceil \cdot \rceil$	rounding up to the nearest integer (ceil function)
$\{\}$	set
$()$	vector
$\dim(\cdot) = \Vert \cdot \Vert_0$	number of elements of a set or number of vector components
$\mathbb{Z}$	set of integer numbers
$\mathbb{Z}_{a,b} = \{a, a+1, \ldots, b-1, b\}$	set of integer numbers from a to b, $a \leq b$
$\mathbb{Z}_{0,\infty} = \{0,1,\ldots\}$	set of non-negative integer numbers

$\mathbb{Z}_{1,\infty} = \mathbb{N}_1 = \{1, 2, \ldots\}$	set of positive integer numbers
$\overline{a} = (a_1, ..., a_n)^T = (a_i : i \in \mathbb{Z}_{1,n})^T$	column vector of length n
$\overline{a}^T = (a_1, ..., a_n) = (a_i : i \in \mathbb{Z}_{1,n})$	row vector of length n
$\mathbb{W}$	matrix
$\mathbb{W}(y, x)$	matrix element, where y denotes a row index and x stands for a column index
$\overline{\mathbb{W}}(y, \cdot)$	row of matrix $\mathbb{W}$ with index y
$\overline{\mathbb{W}}(\cdot, x)$	column of matrix $\mathbb{W}$ with index x
$\mathbb{B}$	Boolean matrix
$H(\mathbb{B})$	number of rows in matrix $\mathbb{B}$
$W(\mathbb{B})$	number of columns in matrix $\mathbb{B}$
$a^{(i)}$	i-th iteration step (state of variable a at i-th iteration step)
$N = 2^n$	number of linear detector cells
C	number of projection images (observed distinct projection angles)
n	exponent of the image linear size (input image is of size $2^n \times 2^n$)
k	exponent of the pattern length (pattern is of length $2^k \leq 2^n$)
I_{2^n}	image of size $2^n \times 2^n$
R_k	tensor of sums over patterns of length 2^k
q	number of ray sums
(x, y)	point or pixel coordinates in the coordinate system $X0Y$
a	shift of the pattern's end (coordinate)
α	number of projection images expressed as a proportion of $C_{max} = 2^{n+2}$
$\Theta(\cdot)$	asymptotic computational complexity (O-symbology)

References

1. Withers, P.J.; Bouman, C.; Carmignato, S.; Cnudde, V.; Grimaldi, D.; Hagen, C.K.; Maire, E.; Manley, M.; Du Plessis, A.; Stock, S.R. X-ray computed tomography. *Nat. Rev. Methods Primers* **2021**, *1*, 18 . [CrossRef]
2. Arlazarov, V.; Nikolaev, D.; Arlazarov, V.; Chukalina, M. X-ray Tomography: The Way from Layer-by-layer Radiography to Computed Tomography. *Comput. Opt.* **2021**, *45*, 897–906. [CrossRef]
3. Gonzalez, S.M. *Interpretation Basics of Cone Beam Computed Tomography*; John Wiley & Sons: Hoboken, NJ, USA, 2021.
4. Kravchenko, D.; Karakostas, P.; Kuetting, D.; Meyer, C.; Brossart, P.; Behning, C.; Schäfer, V.S. The role of dual energy computed tomography in the differentiation of acute gout flares and acute calcium pyrophosphate crystal arthritis. *Clin. Rheumatol.* **2022**, *41*, 223–233. [CrossRef]
5. Sibolt, P.; Andersson, L.M.; Calmels, L.; Sjöström, D.; Bjelkengren, U.; Geertsen, P.; Behrens, C.F. Clinical implementation of artificial intelligence-driven cone-beam computed tomography-guided online adaptive radiotherapy in the pelvic region. *Phys. Imaging Radiat. Oncol.* **2021**, *17*, 1–7. [CrossRef]
6. Carmignato, S.; Dewulf, W.; Leach, R. *Industrial X-ray Computed Tomography*; Springer: Berlin/Heidelberg, Germany, 2018.
7. Silomon, J.; Gluch, J.; Clausner, A.; Paul, J.; Zschech, E. Crack identification and evaluation in BEoL stacks of two different samples utilizing acoustic emission testing and nano X-ray computed tomography. *Microelectron. Reliab.* **2021**, *121*, 114137. . [CrossRef]
8. Potrahov, N.; Anoshkin, A.; Zuiko, V.; Osokin, B.; Pisarev, P.; Pelenev, K. Numerical and experimental study of composite bulkhead partition strength with in-situ X-ray monitoring. *PNRPU Mech. Bull.* **2017**, *1*, 118–133. . perm.mech/2017.1.08. [CrossRef]
9. Kagan, A. Chapter 6—Counterterrorist Detection Techniques of Explosives by Vapor Sensors (Handheld). In *Counterterrorist Detection Techniques of Explosives*, 2nd ed.; Elsevier: Amsterdam, The Netherlands, 2022; pp. 235–251.
10. Champley, K.M.; Willey, T.M.; Kim, H.; Bond, K.; Glenn, S.M.; Smith, J.A.; Kallman, J.S.; Brown, W.D.; Seetho, I.M.; Keene, L.; et al. Livermore tomography tools: Accurate, fast, and flexible software for tomographic science. *NDT E Int.* **2022**, *126*, 102595. . [CrossRef]
11. Hao, S.; Bailey, J.; Iacoviello, F.; Bu, J.; Grant, P.; Brett, D.; Shearing, P. 3D Imaging of Lithium Protrusions in Solid-State Lithium Batteries using X-ray Computed Tomography. *Adv. Funct. Mater.* **2021**, *31*, 2007564. [CrossRef]
12. Grossmann, G. Tomographie. *Fortschr. Röntgenstr.* **1935**, *51*, 61–80.
13. Chaoul, H. Über die Tomographie und insbesondere ihre Anwendung in der Lungendiagnostik. *Fortschr. Röntgenstr.* **1935**, *51*, 342.
14. Inoue, Y.; Itoh, H.; Waga, A.; Sasa, R.; Mitsui, K. Radiation Dose Management in Pediatric Brain CT According to Age and Weight as Continuous Variables. *Tomography* **2022**, *8*, 985–998. [CrossRef]
15. Göppel, S.; Frikel, J.; Haltmeier, M. Feature Reconstruction from Incomplete Tomographic Data without Detour. *Mathematics* **2022**, *10*, 1318. [CrossRef]

16. Mamchur, D.; Peksa, J.; Le Clainche, S.; Vinuesa, R. Application and Advances in Radiographic and Novel Technologies Used for Non-Intrusive Object Inspection. *Sensors* **2022**, *22*, 2121. [CrossRef]
17. Cnudde, V.; Boone, M. High-resolution X-ray computed tomography in geosciences: A review of the current technology and applications. *Earth-Sci. Rev.* **2013**, *123*, 1–17. . [CrossRef]
18. Loterie, D.; Delrot, P.; Moser, C. High-resolution tomographic volumetric additive manufacturing. *Nat. Commun.* **2021**, *11*, 852. [CrossRef]
19. Tanaka, R.; Yoshioka, K.; Takagi, H.; Schuijf, J.; Arakita, K. Novel developments in non-invasive imaging of peripheral arterial disease with CT: Experience with state-of-the-art, ultra-high-resolution CT and subtraction imaging. *Clin. Radiol.* **2019**, *74*, 51–58. [CrossRef]
20. Hata, A.; Yanagawa, M.; Honda, O.; Kikuchi, N.; Miyata, T.; Tsukagoshi, S.; Uranishi, A.; Tomiyama, N. Effect of Matrix Size on the Image Quality of Ultra-high-resolution CT of the Lung: Comparison of 512 × 512, 1024 × 1024, and 2048 × 2048. *Acad. Radiol.* **2018**, *25*, 869–876. [CrossRef]
21. Sun, L.; Yao, J.; Hao, P.; Yang, Y.; Liu, Z.; Peng, R. Diagnostic Role of Four-Dimensional Computed Tomography for Preoperative Parathyroid Localization in Patients with Primary Hyperparathyroidism: A Systematic Review and Meta-Analysis. *Diagnostics* **2021**, *11*, 664. [CrossRef]
22. Wu, D.; Engqvist, J.; Barbier, C.; Karlsson, C.; Hall, S. Unravelling the deformation process of a compacted paper: In-situ tensile loading, 4D X-ray tomography and image-based analysis. *Int. J. Solids Struct.* **2022**, *242*, 111539. . [CrossRef]
23. Buzug, T.M. Computed tomography. In *Springer Handbook of Medical Technology*; Springer: Berlin/Heidelberg, Germany, 2011; pp. 311–342.
24. Hatcher, D.C. Operational Principles for Cone-Beam Computed Tomography. *J. Am. Dent. Assoc.* **2010**, *141*, 3S–6S. [CrossRef]
25. Gong, H.; Tao, S.; Gagneur, D.; Liu, W.; Shen, J.; McCollogh, C.; Hu, Y.; Leng, S. Implementation and experimental evaluation of Mega-voltage fan-beam CT using a linear accelerator. *Radiat. Oncol.* **2021**, *16*, 139. [CrossRef]
26. Yoneyama, A.; Baba, R.; Hyodo, K.; Takeda, T.; Nakano, H.; Maki, K.; Sumitani, K.; Hirai, Y. Development of high-resolution X-ray CT system using parallel beam geometry. *AIP Conf. Proc.* **2016**, *1696*, 020007. [CrossRef]
27. Grass, M.; Kohler, T.; Proksa, R. 3D cone-beam CT reconstruction for circular trajectories. *Phys. Med. Biol.* **2000**, *45*, 329–348. [CrossRef]
28. Heiken, J.; Brink, A.; Vannier, M. Spiral (helical) CT. *Radiology* **1993**, *189*, 647–656. . [CrossRef]
29. Valton, S.; Peyrin, F.; Sappey-Marinier, D. A FDK-Based Reconstruction Method for Off-Centered Circular Trajectory Cone Beam Tomography. *IEEE Trans. Nucl. Sci.* **2006**, *53*, 2736–2745. [CrossRef]
30. Kulkarni, S.; Rumberger, J.; Jha, S. Electron Beam CT: A Historical Review. *Am. J. Roentgenol.* **2021**, *216*, 1222–1228. [CrossRef]
31. Nikitin, V. TomocuPy—Efficient GPU-based tomographic reconstruction with asynchronous data processing. *J. Synchrotron Radiat.* **2023**, *30*, 179–191. [CrossRef]
32. Nourazar, M.; Goossens, B. Accelerating iterative CT reconstruction algorithms using Tensor Cores. *J. Real-Time Image Process.* **2021**, *18*, 1979–1991. [CrossRef]
33. Wang, X.; Zhang, Y.; Hong, X.; Wang, H. Fast Backprojection Filtration Algorithm in Circular Cone-Beam Computed Tomography. *Int. J. Opt.* **2023**, *2023*, 1749624. [CrossRef]
34. Zhang, S.; Zhang, Y.; Tuo, M.; Zhang, H. Fast algorithm for Joseph's forward projection in iterative computed tomography reconstruction. *J. Ambient. Intell. Humaniz. Comput.* **2023**, *14*, 12535–12548. [CrossRef]
35. Bulatov, K.B.; Chukalina, M.V.; Nikolaev, D.P. *Fast X-ray Sum Calculation Algorithm for Computed Tomography Problem*; Series of Mathematical Modelling, Programming & Computer Software; Bulletin of the South Ural State University: Chelyabinsk, Russia, 2020; Volume 13.
36. Feldkamp, L.A.; Davis, L.C.; Kress, J.W. Practical cone-beam algorithm. *J. Opt. Soc. Am. A* **1984**, *1*, 612–619. [CrossRef]
37. Mersereau, R.; Oppenheim, A. Digital reconstruction of multidimensional signals from their projections. *Proc. IEEE* **1974**, *62*, 1319–1338. [CrossRef]
38. Natterer, F. *The Mathematics of Computerized Tomography*; Mir: Moscow, Russia, 1990; p. 288.
39. Dusaussoy, N. VOIR: A volumetric image reconstruction algorithm based on Fourier techniques for inversion of the 3-D Radon transform. *IEEE Trans. Med. Process.* **1996**, *5*, 121–131. [CrossRef]
40. Averbuch, A.; Coifman, R.; Donoho, D.; Israeli, M.; Waldén, J. Fast slant stack: A notion of Radon transform for data in a Cartesian grid which is rapidly computible, algebraically exact, geometrically faithful and invertible. *SIAM J. Sci. Comput* **2001**, *37*, 192–206.
41. Matej, S.; Fessler, J.; Kazantsev, I. Iterative tomographic image reconstruction using Fourier-based forward and back-projectors. *IEEE Trans. Med. Imaging* **2004**, *23*, 401–412. [CrossRef]
42. Sullivan, J. A fast sinc function gridding algorithm for fourier inversion in computer tomography. *IEEE Trans. Med. Imaging* **1985**, *4*, 200–207. [CrossRef]
43. Arcadu, F.; Nilchian, M.; Studer, A.; Stampanoni, M.; Marone, F. A Forward Regridding Method with Minimal Oversampling for Accurate and Efficient Iterative Tomographic Algorithms. *IEEE Trans. Med. Process.* **2016**, *23*, 1207–1218. [CrossRef]
44. Zhang, Z.; Han, X.; Pearson, E.; Pelizzari, C.; Sidky, E.; Pan, X. Artifact reduction in short-scan CBCT by use of optimization-based reconstruction. *Phys. Med. Biol.* **2016**, *61*, 3387–3406. [CrossRef]

45. Brady, M.L.; Yong, W. Fast Parallel Discrete Approximation Algorithms for the Radon Transform. In Proceedings of the SPAA '92, Fourth Annual ACM Symposium on Parallel Algorithms and Architectures, San Diego, CA, USA, 29 June–1 July 1992; pp. 91–99. [CrossRef]
46. Götz, W.A.; Druckmüller, H.J. A fast digital Radon transform—An efficient means for evaluating the Hough transform. *Pattern Recognit.* **1995**, *28*, 1985–1992. [CrossRef]
47. Basu, S.; Bresler, Y. O(N/sup 2/log/sub 2/N) filtered backprojection reconstruction algorithm for tomography. *IEEE Trans. Image Process.* **2000**, *9*, 1760–1773. [CrossRef]
48. Xiao, S.; Bresler, Y.; Munson, D. O(N/sup 2/ log N) native fan-beam tomographic reconstruction. In Proceedings of the IEEE International Symposium on Biomedical Imaging, Washington, DC, USA, 7–10 July 2002; pp. 824–827. [CrossRef]
49. Xiao, S.; Bresler, Y.; Munson, D. Fast Feldkamp algorithm for cone-beam computer tomography. In Proceedings of the 2003 International Conference on Image Processing (Cat. No.03CH37429), Barcelona, Spain, 14–17 September 2003; p. II-819. [CrossRef]
50. Brokish, J.; Bresler, Y. Ultra-fast hierarchical backprojection for Micro-CT reconstruction. In Proceedings of the IEEE Nuclear Science Symposium Conference Record, Honolulu, HI, USA, 26 October–3 November 2007; pp. 4460–4463. [CrossRef]
51. Bresler, Y.; Brokish, J. A hierarchical algorithm for fast backprojection in helical cone-beam tomography. In Proceedings of the 2nd IEEE International Symposium on Biomedical Imaging: Nano to Macro (IEEE Cat No. 04EX821), Arlington, VA, USA, 18 April 2004; pp. 1420–1423. [CrossRef]
52. Brady, M.L. A fast discrete approximation algorithm for the Radon transform. *SIAM J. Comput.* **1998**, *27*, 91–99. [CrossRef]
53. Ershov, E.; Terekhin, A.; Nikolaev, D.; Postnikov, V.; Karpenko, S. Fast Hough transform analysis: Pattern deviation from line segment. In Proceedings of the SPIE 9875, Eighth International Conference on Machine Vision (ICMV 2015), Barcelona, Spain, 19–21 November 2015; Volume 9875, pp. 42–46. [CrossRef]
54. Karpenko, S.M.; Ershov, E.I. Analysis of Properties of Dyadic Patterns for the Fast Hough Transform. *Probl. Inf. Transm.* **2021**, *57*, 292–300. [CrossRef]
55. Nikolaev, D.; Ershov, E.; Kroshnin, A.; Limonova, E.; Mukovozov, A.; Faradzhev, I. On a Fast Hough/Radon Transform as a Compact Summation Scheme over Digital Straight Line Segments. *Mathematics* **2023**, *11*, 3336. [CrossRef]
56. Wu, T.K.; Brady, M.L. A fast approximation algorithm for 3D image reconstruction. In Proceedings of the 1998 International Computer Symposium. Workshop in Image Processing and Character Recognition, Bombay, India, 4–7 January 1998; pp. 213–220.
57. Ershov, E.I.; Terekhin, A.P.; Nikolaev, D.P. Fast Hough transform generalization for three-dimensional images. *Inf. Process.* **2017**, *17*, 294–308. (In Russian). [CrossRef]
58. Radon, J. On the determination of functions from their integral values along certain manifolds. *IEEE Trans. Med. Imaging* **1986**, *5*, 170–176. [CrossRef]
59. Ingacheva, A.S.; Chukalina, M.V. Polychromatic CT data improvement with one-parameter power correction. *Math. Probl. Eng.* **2019**, *2019*, 1405365. [CrossRef]
60. Chukalina, M.V.; Buzmakov, A.V.; Ingacheva, A.S.; Shabelnikova, Y.L.; Vsadchikov, V.E.; Bukreeva, I.N.; Nikolaev, D.P. Analysis of the tomographic reconstruction from polychromatic projections for objects with highly absorbing inclusions. *Inf. Technol. Comput. Syst.* **2020**, 3, 49–61. [CrossRef]
61. Hammersberg, P.; Mångård, M. Correction for beam hardening artefacts in computerized tomography. *J. X-ray Sci. Technol.* **1998**, *8*, 75–93.
62. Reiter, M.; de Oliveira, F.B.; Bartscher, M.; Gusenbauer, C.; Kastner, J. Case study of empirical beam hardening correction methods for dimensional X-ray computed tomography using a dedicated multi-material reference standard. *J. Nondestruct. Eval.* **2019**, *38*, 1–15. [CrossRef]
63. Lewitt, R.M. Reconstruction algorithms: Transform methods. *Proc. IEEE* **1983**, *71*, 390–408. [CrossRef]
64. Horbelt, S.; Liebling, M.; Unser, M.A. Filter design for filtered backprojection guided by the interpolation model. In *Medical Imaging 2002: Image Processing, Proceedings of the Medical Imaging 2002, San Diego, CA, USA, 23–28 February 2002*; SPIE: Bellingham, WA, USA, 2002; Volume 4684, pp. 806–813.
65. Myagotin, A.; Voropaev, A.; Helfen, L.; Hänschke, D.; Baumbach, T. Efficient volume reconstruction for parallel-beam computed laminography by filtered backprojection on multi-core clusters. *IEEE Trans. Image Process.* **2013**, *22*, 5348–5361. [CrossRef]
66. La Riviere, P.J.; Pan, X. Comparison of angular interpolation approaches in few-view tomography using statistical hypothesis testing. In *Medical Imaging 1999: Image Processing, Proceedings of the Medical Imaging 1999, San Diego, CA, USA, 20–26 February 1999*; SPIE: Bellingham, WA, USA, 1999; Volume 3661, pp. 398–407.
67. Mileto, A.; Guimaraes, L.S.; McCollough, C.H.; Fletcher, J.G.; Yu, L. State of the art in abdominal CT: The limits of iterative reconstruction algorithms. *Radiology* **2019**, *293*, 491–503. [CrossRef]
68. Gordon, R.; Bender, R.; Herman, G.T. Algebraic reconstruction techniques (ART) for three-dimensional electron microscopy and X-ray photography. *J. Theor. Biol.* **1970**, *29*, 471–481. [CrossRef]
69. Gilbert, P. Iterative methods for the three-dimensional reconstruction of an object from projections. *J. Theor. Biol.* **1972**, *36*, 105–117. [CrossRef]
70. Herman, G.T.; Meyer, L.B. Algebraic reconstruction techniques can be made computationally efficient (positron emission tomography application). *IEEE Trans. Med. Imaging* **1993**, *12*, 600–609. [CrossRef]
71. Sidky, E.Y.; Pan, X. Image reconstruction in circular cone-beam computed tomography by constrained, total-variation minimization. *Phys. Med. Biol.* **2008**, *53*, 4777. [CrossRef]

72. Satzoda, R.K.; Suchitra, S.; Srikanthan, T. Parallelizing the Hough transform computation. *IEEE Signal Process. Lett.* **2008**, *15*, 297–300. [CrossRef]
73. Kerr, J.P.; Bartlett, E.B. Neural network reconstruction of single-photon emission computed tomography images. *J. Digit. Imaging* **1995**, *8*, 116–126. [CrossRef]
74. Xie, H.; Shan, H.; Wang, G. Deep encoder-decoder adversarial reconstruction (DEAR) network for 3D CT from few-view data. *Bioengineering* **2019**, *6*, 111. [CrossRef]
75. Ma, G.; Zhang, Y.; Zhao, X.; Wang, T.; Li, H. A neural network with encoded visible edge prior for limited-angle computed tomography reconstruction. *Med. Phys.* **2021**, *48*, 6464–6481. [CrossRef]
76. Wang, W.; Xia, X.G.; He, C.; Ren, Z.; Lu, J. A model-based deep network for limited-angle computed tomography image reconstruction. *Displays* **2022**, *73*, 102166. [CrossRef]
77. Yamaev, A.; Chukalina, M.; Nikolaev, D.; Sheshkus, A.; Chulichkov, A. Lightweight denoising filtering neural network for FBP algorithm. In *Proceedings of the Thirteenth International Conference on Machine Vision, Rome, Italy, 2–6 November 2020*; SPIE: Bellingham, WA, USA, 2021; Volume 1165, p. 116050L. [CrossRef]
78. Götz, W. Eine Schnelle Diskrete Radon-Transformation Basierend auf Rekursiv Definierten Digitalen Geraden. Ph.D. Thesis, University of Innsbruck, Innsbruck, Austria, 1993.
79. Vuillemin, J.E. Fast linear Hough transform. In Proceedings of the IEEE International Conference on Application Specific Array Processors (ASSAP'94), San Francisco, CA, USA, 22–24 August 1994; pp. 1–9.
80. Frederick, M.T.; VanderHorn, N.A.; Somani, A.K. Real-time H/W implementation of the approximate discrete Radon transform. In Proceedings of the 2005 IEEE International Conference on Application-Specific Systems, Architecture Processors (ASAP'05), Samos, Greece, 23–25 July 2005; pp. 399–404.
81. Nikolaev, D.P.; Karpenko, S.M.; Nikolaev, I.P.; Nikolayev, P.P. Hough transform: Underestimated tool in the computer vision field. In Proceedings of the ECMS 2008, Nicosia, Cyprus, 3–6 June 2008; pp. 238–243. [CrossRef]
82. Ershov, E.I.; Karpenko, S. Fast Hough Transform and approximation properties of dyadic patterns. *arXiv* **2013**, arXiv:1712.05615.
83. Aliev, M.; Ershov, E.I.; Nikolaev, D.P. On the use of FHT, its modification for practical applications and the structure of Hough image. In Proceedings of the ICMV 2018, Munich, Germany, 1–3 November 2018; Volume 11041. [CrossRef]
84. Prun, V.; Nikolaev, D.; Buzmakov, A.; Chukalina, M.; Asadchikov, V. Effective regularized algebraic reconstruction technique for computed tomography. *Crystallogr. Rep.* **2013**, *58*, 1063–1066. [CrossRef]
85. Andersen, A.H.; Kak, A.C. Simultaneous algebraic reconstruction technique (SART): A superior implementation of the ART algorithm. *Ultrason. Imaging* **1984**, *6*, 81–94. [CrossRef]
86. Dolmatova, A.; Chukalina, M.; Nikolaev, D. Accelerated FBP for computed tomography image reconstruction. In Proceedings of the IEEE ICIP 2020, Washington, DC, USA, 26–27 October 2020; Number CIS-02.2, pp. 3030–3034. . [CrossRef]
87. Ershov, E.; Terekhin, A.; Nikolaev, D. Generalization of the fast hough transform for three-dimensional images. *J. Commun. Technol. Electron.* **2018**, *63*, 626–636. [CrossRef]
88. Sidky, E.Y.; Kao, C.M.; Pan, X. Accurate image reconstruction from few-views and limited-angle data in divergent-beam CT. *J. X-ray Sci. Technol.* **2006**, *14*, 119–139.
89. Bulatov, K.; Chukalina, M.; Buzmakov, A.; Nikolaev, D.; Arlazarov, V.V. Monitored Reconstruction: Computed Tomography as an Anytime Algorithm. *IEEE Access* **2020**, *8*, 110759–110774. [CrossRef]
90. CT Software Smart Tomo Engine. Available online: https://smartengines.com/ocr-engines/tomo-engine/ (accessed on 22 November 2023).
91. Wang, Z.; Bovik, A.C.; Sheikh, H.R.; Simoncelli, E.P. Image quality assessment: From error visibility to structural similarity. *IEEE Trans. Image Process.* **2004**, *13*, 600–612. [CrossRef]
92. Kruskal, J.B. Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika* **1964**, *29*, 1–27. [CrossRef]