Industrial Computed Tomography using Proximal Algorithm

Thesis by

Guangming Zang

In Partial Fulfillment of the Requirements

For the Degree of

Masters of Science

King Abdullah University of Science and Technology, Thuwal,
Kingdom of Saudi Arabia

April, 2016
The thesis of Guangming Zang is approved by the examination committee

Committee Chairperson: Peter Wonka
Committee Co-Chairperson: Wolfgang Heidrich
Committee Member: Ganesh Sundaramoorthi
ABSTRACT

Industrial Computed Tomography using Proximal Algorithm

Guangming Zang

In this thesis, we present ProxiSART, a flexible proximal framework for robust 3D cone beam tomographic reconstruction based on the Simultaneous Algebraic Reconstruction Technique (SART). We derive the proximal operator for the SART algorithm and use it for minimizing the data term in a proximal algorithm. We show the flexibility of the framework by plugging in different powerful regularizers, and show its robustness in achieving better reconstruction results in the presence of noise and using fewer projections. We compare our framework to state-of-the-art methods and existing popular software tomography reconstruction packages, on both synthetic and real datasets, and show superior reconstruction quality, especially from noisy data and a small number of projections.
ACKNOWLEDGEMENTS

I would like to thank my supervisor, Dr. Peter Wonka, and my co-supervisor, Dr. Wolfgang Heidrich, for their endless guidance and support during my research and study at KAUST. I have been extremely lucky to have two supervisors who were always there to help me, and who taught me how to do interesting research and showed their great patience in my studies.

I would like to thank the faculty members of VCC, especially Dr. Ganesh Sundaramoorthi for joining my thesis committee. I would also like to thank Dr. Mohamed Aly for his help, suggestions and comments during my thesis, and all members of the Visual Computing Center and KAUST for their support.

Finally, I would like to thank my family, my wife and her family for their constant support and unconditional love.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Examination Committee Approval</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Copyright</td>
<td>3</td>
</tr>
<tr>
<td>Abstract</td>
<td>4</td>
</tr>
<tr>
<td>Acknowledgements</td>
<td>5</td>
</tr>
<tr>
<td>List of Figures</td>
<td>8</td>
</tr>
<tr>
<td>1 Introduction</td>
<td>10</td>
</tr>
<tr>
<td>1.1 Related Work</td>
<td>13</td>
</tr>
<tr>
<td>1.2 Contributions of This Thesis</td>
<td>15</td>
</tr>
<tr>
<td>1.3 Organization</td>
<td>15</td>
</tr>
<tr>
<td>2 Methods for CT Cone Beam Reconstruction</td>
<td>17</td>
</tr>
<tr>
<td>2.1 Geometry for cone beam reconstruction</td>
<td>17</td>
</tr>
<tr>
<td>2.2 Simultaneous Algebraic Reconstruction Technique (SART)</td>
<td>18</td>
</tr>
<tr>
<td>2.2.1 Forward projection</td>
<td>19</td>
</tr>
<tr>
<td>2.2.2 Correction</td>
<td>19</td>
</tr>
<tr>
<td>2.2.3 Backprojection</td>
<td>19</td>
</tr>
<tr>
<td>2.3 The Reconstruction ToolKit (RTK)</td>
<td>21</td>
</tr>
<tr>
<td>3 Proximal Algorithms for Volume Reconstruction</td>
<td>22</td>
</tr>
<tr>
<td>3.1 Inverse Problem</td>
<td>22</td>
</tr>
<tr>
<td>3.2 Proximal Algorithm</td>
<td>23</td>
</tr>
<tr>
<td>3.2.1 Alternating Direction Method of Multipliers (ADMM)</td>
<td>24</td>
</tr>
<tr>
<td>3.2.2 Linearized ADMM</td>
<td>25</td>
</tr>
<tr>
<td>3.2.3 Chambolle-Pock Primal-Dual Algorithm</td>
<td>28</td>
</tr>
<tr>
<td>4 ProxiSART Framework</td>
<td>29</td>
</tr>
<tr>
<td>4.1 Data Fidelity Operator</td>
<td>31</td>
</tr>
</tbody>
</table>
LIST OF FIGURES

1.1 Volume rendering results for the Shepp-Logan phantom (left), the RabbitCT (center), and the Lynx dataset (right). .................................................. 11
1.2 Nikon XT 225 industrial computed tomography scanner. .................. 13
2.1 Geometry for 3D cone beam CT reconstruction. .............................. 18
3.1 Forward problems and inverse problems. ........................................ 23
5.1 Effect of the number of projections showing SNR (left) and SSIM (right) as a function of the number of projections used using Shepp-Logan phantom. .................................................. 51
5.2 Effect of noise showing SNR (left) and SSIM (right) as a function of the amount of injected noise using Shepp-Logan phantom. ................. 51
5.3 Speed of convergence: SNR as a function of the number of iterations for the Shepp-Logan phantom with 60 projections and $\sigma = 0.75$ Gaussian noise. ................................................................. 52
5.4 A sample slice with different number of projections for different algorithms. From top to bottom: 90, 60, 45, and 30 projections. From left to right: RTK-FDK, RTK-SART, PlainSART, RTK-ADMM, PSART-ITV, PSART-SAD. The displayed range of values is $[0.97,1.09]$. .......... 52
5.5 A sample slice with amounts of noise for different algorithms. From top to bottom: standard deviations of 0.05, 0.2, 0.5, and 0.75. From left to right: RTK-FDK, RTK-SART, PlainSART, RTK-ADMM, PSART-ITV, PSART-SAD. The displayed range of values is $[0.97,1.09]$. .......... 53
5.6 Sample slices for the reconstruction of the RabbitCT dataset in the Sagittal (top), Axial (middle), and Coronal (bottom) planes. From left to right: RTK-FDK, RTK-SART, PlainSART, RTK-ADMM, PSART-ITV, PSART-SAD. ................................................................. 55
Sample slices for the reconstruction of the Lynx dataset in the Sagittal (top), Axial (middle), and Coronal (bottom) planes. From left to right: RTK-FDK, RTK-SART, PlainSART, RTK-ADMM, PSART-ITV, PSART-SAD.
Chapter 1

Introduction

Reducing the dosage in X-ray tomography is a very important issue in medical application, since long term exposure to X-rays can have adverse health effects. This can be done in at least two ways: (a) reducing the X-ray beam power, which leads to increased measurement noise at the detectors; or (b) acquiring fewer projections to reduce the acquisition time\[1\]. This makes the reconstruction problem even more ill-posed, since less information is collected from the volume to be reconstructed; and one has to use non-linear regularizers (priors) to achieve a reasonable result. This is typically done using iterative solvers \[2 \ 3\].

Iterative algorithms for X-ray tomography reconstruction have been around for years. In fact, one of the first implemented tomography reconstruction algorithm was an iterative one, the Algebraic Reconstruction Technique (ART) \[4 \ 5 \ 6\]. However, non-iterative transform-based algorithms, such as the filtered back projection \[7 \ 8 \ 9 \ 10\], have been more popular due to their speed and low computational cost. In fact, most commercial X-ray CT scanners employ some variant of FBP in their reconstruction software \[11\]. Recently, interest has been ignited again in iterative algorithms because, although they are more computationally demanding, they are much more flexible and usually yield superior reconstruction quality by employing powerful priors. The iterative algorithms used in tomography have to be “matrix free” i.e. need to work without explicitly storing the system matrix in memory (see
Figure 1.1: Volume rendering results for the Shepp-Logan phantom (left), the RabbitCT (center), and the Lynx dataset (right).

Eq. 1.1 since this requires huge amounts of memory. For example, with a $512^3$ reconstruction volume and using 360 1.5-megapixel projections, the system matrix has about 70 billion entries i.e. would require 280 GB of memory using single-precision floating point numbers.

We focus on iterative reconstruction techniques, and especially the Simultaneous Algebraic Reconstruction Technique (SART) [12, 13]. SART has been shown to provide better reconstruction quality, and is amenable to parallelization using graphics hardware [14, 15]. It also has the advantage of having an intuitive geometric interpretation for the update equation (see Eq. 2.1). SART is based on ART, an iterative method dating back to Kaczmarz in 1937 [16]. It solves a linear system of the form

$$ Ax = p $$

(1.1)

where $x \in \mathbb{R}^n$ is the unknown 3D reconstruction volume in vector form, $A \in \mathbb{R}^{m \times n}$ is the system matrix, and $p \in \mathbb{R}^m$ represents the measured line projections (sinogram). It is a kind of projection method where the the new estimate is computed as

$$ x^{(t+1)} = x^{(t)} + \Delta x^{(t)} $$
where \( x^{(k)} \) is the solution at the \( t^{th} \) iteration and \( \Delta x^{(t)} \) is the correction term for the \( t^{th} \) iteration computed from the projection of the current estimate on the hyperplane defined by one of the equations of the linear system. It has been shown that ART computes a solution \( x^* \) of minimum norm to the linear system above \([17]\). In SART, instead of performing the update for each row of \( A \), the updates of a subset of the rows are aggregated and applied simultaneously at the same time. This provides more stability against noise and faster convergence for SART \([15, 13]\). This is also related to another class of algorithms called Ordered Subset algorithms \([18, 19, 20, 21]\).

Proximal algorithms are a class of optimization algorithms that are quite flexible and powerful \([22, 23, 24]\). They are generally used to efficiently solve non-smooth, constrained, distributed, or large scale optimization problems. They are more modular than other optimization problems, in the sense that they provide a few lines of code that depend on solving smaller conventional, and usually simpler, optimization problems called proximal operator. The proximal operator \([25, 22, 24]\) for a function \( h(\cdot) \) is a generalization of projections on convex sets, and can be thought of intuitively as getting closer to the optimal solution while staying close to the current estimate. Formally it is defined as

\[
\text{prox}_{\lambda h}(u) = \arg\min_x h(x) + \frac{1}{2\lambda} \|x - u\|_2^2
\]

where \( x, u \in \mathbb{R}^n \) and \( \lambda \) is a regularization parameter. Many proximal operators of common functions are easy to compute, and often admit a closed form solution. Computing the proximal operator of a certain function opens the way to solving hard optimization problems involving this function and other regularization terms e.g. smoothing norms or sparsity inducing norms, which otherwise is not generally easy.
1.1 Related Work

X-ray tomography reconstruction has received extensive attention since the first practical medical CT device was invented in the early 1970s by Hounsfield [26, 27]. There are two general approaches for tomography reconstruction: transform-based methods and iterative methods [4, 1]. Transform methods rely on the Radon transform and its inverse introduced in 1917. The most widely used 3D cone beam reconstruction method is the filtered backprojection algorithm introduced by Feldkamp, Davis, and Kress and known as FDK [9]. Transform methods are usually viewed as much faster than iterative methods, and have therefore been the method of choice for X-ray scanner manufacturers [11]. For instance, Nikon XT series scanners, as shown in Figure 1.2.

Iterative methods on the other hand use algebraic techniques to solve the reconstruction problem. They generally model the problem as a linear system of the form in Eq. 1.1 and established numerical linear algebraic methods [1]. The main challenge in computed tomography, and especially for cone beam reconstruction, is the huge memory required to store the system matrix. This puts a limit on the range of algo-
gorithms that can be used efficiently. ART and its many variants are among the best known iterative reconstruction algorithms \[5, 28, 29, 30, 12, 13\]. They use variations of the projection method of Kaczmarz and have modest memory requirements, and have been shown to yield better reconstruction results than transform methods. They are matrix free, and work without having to explicitly store the system matrix.

In addition, iterative methods provide more flexibility in incorporating prior information into the reconstruction process. For example, instead of assuming a Gaussian noise model and minimizing a least squares data term, one can easily use iterative methods with a Poisson noise model \[31, 32, 33, 34, 2\]. The data term leads to a weighted least squares problem instead, with weights proportional to the detected intensities. Priors are also easy to use with iterative methods. For example, the Total Variation \[35\] prior has been used for tomography reconstruction \[36, 37\].

Proximal algorithms have been widely used in many problems in machine learning and signal processing \[25, 22, 23, 24\]. In particular, they have also been used in tomography reconstruction. For example, \[37\] used the Alternating Direction Method of Multipliers (ADMM) \[23\] with total variation prior, where the data term was optimized using Conjugate Gradient (CG) \[38\]. \[39\] discussed using the Chambolle-Pock algorithm \[40\] for tomography reconstruction with different priors. \[41\] used ADMM with Preconditioned Conjugate Gradient \[38\] for optimizing the weighted least squares data term. \[21\] used Linearized ADMM \[24\] (also known as Inexact Split Uzawa \[42\]) with Ordered Subset-based methods \[19\] for optimizing the data term and FISTA \[43\] for optimizing the prior term. However, none of these methods used SART as their data term solver, which has several advantages over other methods in terms of memory requirements and the quality of reconstruction. Moreover, CG is known for being sensitive to noise, and as we will show in the experiments (see Sec. \[5\]) is not as robust as SART.

There are currently a number of open source software packages for tomography
reconstruction. SNARK09 [44] is one of the oldest. It has several algorithms implemented for 2D reconstruction, but very little support for 3D reconstruction. The Reconstruction ToolKit (RTK) [45] is a high performance C++ toolkit focusing on 3D cone beam reconstruction that is based on the image processing package Insight ToolKit (ITK). It includes implementations of several algorithms, including FDK, SART, and an ADMM TV-regularized solver with CG [37]. The ASTRA toolbox [46] is a Matlab-based GPU-accelerated toolbox for tomography reconstruction. It includes implementations of several algorithms, including SART, SIRT, FDK, FBP, among others. However, it doesn’t have SART for 3D reconstruction.

1.2 Contributions of This Thesis

In this work we make the following contributions:

1. We derive the proximal operator for the SART algorithm.

2. We present ProxiSART, a proximal framework for robust tomography reconstruction based on SART, that has the flexibility of working with many powerful regularizers, including standard ones e.g. Total Variation (TV), and non-standard ones e.g. Sum of Absolute Differences (SAD).

3. We validate the efficacy of our algorithms and show superior reconstruction quality compared to existing popular methods and software packages, in the presence of measurement noise and using fewer projections.

1.3 Organization

This thesis is organized as follows:

Chapter 2 gives a brief introduction into reconstruction algorithms used in cone beam CT, both transform methods and iterative methods. In addition, a description
of RTK, an open source software packages for tomography reconstruction, will be given.

Chapter 3 gives some basic terminology about inverse problems and describes proximal algorithms, their parameter settings and some applications based on them.

Chapter 4 describes a proximal framework called ProxiSART for robust tomography reconstruction based on SART, showing how flexible it is to plug in different novel and powerful regularizers based on our prior information.

Chapter 5 gives a comprehensive comparison between our framework and state-of-the-art methods and existing popular software tomography reconstruction packages, on both synthetic and real datasets, and shows superior reconstruction quality.

Finally, Chapter 6 concludes the work introduced in this thesis, and future work is also explored in this chapter.
Chapter 2

Methods for CT Cone Beam Reconstruction

2.1 Geometry for cone beam reconstruction

In this section, we describe geometry and parameters settings of 3D cone beam CT reconstruction. As shown in Figure 2.1 to reconstruct a 3D volume, a series of projection images is required. For each projection image, we need the X ray source position, object position, and detector position in world coordinates. In addition, the source ray to iso-object distance (sid), and the source ray to detector distance are also very important geometry parameters for volume reconstruction. In practice, we need multiple matrices to represent different transformations in different coordinate systems:

- Voxel2VolumeMatrix: The transformation from logical voxel index to volume physical position.

- Volume2WorldMatrix: The transformation from volume coordination to world coordinates.

- World2ViewMatrix: The transformation from world coordination to view (camera) coordinates.
2.2 Simultaneous Algebraic Reconstruction Technique (SART)

We begin by describing the SART algorithm [12, 13, 14], see Algorithm 1. SART is an iterative algorithm, where at each iteration the current estimate of the reconstructed volume is updated based on how well it fits the input projections. In particular, the update equation for each voxel $x_j$ in the volume $x$ is:

$$x_j^{(t+1)} = x_j^{(t)} + \alpha \sum_{i \in S} c_i^{(t)} a_{ij} \sum_{i \in S} a_{ij}$$  \hspace{1cm} (2.1)$$

where

$$c_i^{(t)} = \frac{p_i - \hat{p}_i^{(t)}}{\sum_k a_{ik}}$$  \hspace{1cm} (2.2)$$

- View2PixelMatrix: The transformation from view (camera) coordinates to image pixel (or device) coordinates.

Figure 2.1: Geometry for 3D cone beam CT reconstruction.
is the normalized correction factor for ray $i$ that measures the residual between the measured projection value $p_i$ and the current estimate at iteration $t$:

$$\hat{p}_{i}^{(t)} = \sum_{k} a_{ik} x_{k}^{(t)} \quad (2.3)$$

$\alpha$ is a relaxation parameter usually $0 < \alpha < 2$, $S$ is a set of projection rays under consideration, and $a_{ij}$ is the element in row $i$ and column $j$ of the system matrix $A$ and defines the contribution to ray sum $i$ from voxel $j$. Basically the equation can be decomposed into three steps [14]:

### 2.2.1 Forward projection

In forward projection stage, SART computes the estimated projection $\hat{p}_{i}^{(t)}$ for each ray $i$ from the current volume $x^{(t)}$ (Equation 2.3). This corresponds to a volume rendering operation.

### 2.2.2 Correction

In this stage, SART computes $c_{i}^{(t)}$, the normalized deviation of this estimate from the true projection $p_i$, where the correction is normalized by the contribution of this ray to all the voxels it goes through (Eq 2.2).

### 2.2.3 Backprojection

In backprojection stage, this correction factor is distributed back to all the voxels that contribute to this ray sum (Eq. 2.1).

SART is an improvement to the original ART algorithm [5, 6]. Unlike ART where a single iteration processes only one projection line $p_i$ and performs the three steps above, SART combines the projection lines from a single projection image and applies the correction simultaneously. This provides faster convergence, better noise
resilience, and faster computations \cite{13, 14}. In the case where the linear system is under-determined i.e. there are fewer equations than unknowns \((m < n)\), ART has been shown to converge to a minimum norm solution of the linear system describing the tomography problem \cite{17}. In particular, the solution provided by ART is equivalent to the optimal solution of the following problem

\[
\min_x \|x\|_2^2 \quad \text{subject to} \quad Ax = p.
\] (2.4)

There has been no convergence proof for SART, although there have been proofs for a simplified version of SART \cite{47, 48} that establish its convergence to a weighted least squares solution. In this work, we use the original SART, and assume it solves the original problem as ART i.e. it solves the minimum norm problem as defined in Eq.2.4.

\begin{algorithm}
\caption{Standard SART}
\begin{algorithmic}
\Require: \(A \in \mathbb{R}^{m \times n}, \alpha \in \mathbb{R}, p \in \mathbb{R}^m\)
\State 1: Initialize: \(x^{(0)} = 0\)
\State 2: \textbf{for all} \(t = 1 \ldots T\) \textbf{do}
\State 3: \qquad \textbf{for all} projection images \(S \in S_1 \ldots S_N\) \textbf{do}
\State 4: \qquad \quad \hat{p}_i^{(t)} = \sum_k a_{ik} x_k^{(t)}
\State 5: \qquad \quad c_i^{(t)} = \frac{p_i - \hat{p}_i^{(t)}}{\sum_k a_{ik}}
\State 6: \qquad \quad x_j^{(t+1)} = x_j^{(t)} + \alpha \frac{\sum_{i \in S} c_i^{(t)} a_{ij}}{\sum_{i \in S} a_{ij}} \quad \text{for} \quad j = 1 \ldots n
\State 7: \qquad \textbf{end for}
\State 8: \textbf{end for}
\State 9: \Return volume reconstruction \(x \in \mathbb{R}^n\)
\end{algorithmic}
\end{algorithm}
2.3 The Reconstruction ToolKit (RTK)

RTK is a high performance toolkit focusing on 3D cone beam reconstruction, which is based on the image processing package Insight ToolKit (ITK). RTK includes C++ implementations of several popular algorithms, such as FDK, SART, and an alternating direction method of multipliers (ADMM) total variation-regularized solver with conjugate gradient (CG) method. What’s more, the geometry description in RTK is identical to what we used in our framework described in next chapter, making it possible to make a comprehensive comparison between our proposed methods and state of art algorithms implemented in RTK.
Chapter 3

Proximal Algorithms for Volume Reconstruction

3.1 Inverse Problem

Inverse problems are not only an exciting and expanding area of mathematics, but also a rich source of theoretical and applied problems as shown in Figure 3.1. All CT reconstruction methods described in this thesis are typical inverse problems.

- Forward problem
  The forward problem (i.e., direct problem) describes the case in which the cause and parameters are given, how to find the effect and observation. i.e., based on the model, how to result in the observation. For instance, in a volume rendering process, when we know how X-rays waves attenuate as they pass through an object in response to its composition.

- Inverse problem
  An inverse problem, on the other hand, describes how to estimate the causes and parameters from given effect and observations. For instance, in the volume reconstruction process, given the attenuated X-ray radiation, how to estimate the material composition of the object.
3.2 Proximal Algorithm

Alternating Direction Method of Multipliers (ADMM) is a technique to solve minimization problems of the form

$$\min_x f(x) + g(x)$$

with variable $x$ and objectives $f(x)$ and $g(x)$ that are separable. For example, $f(x)$ can be a data term that measures the goodness of fit of the current solution, while $g(x)$ can be the prior term that measures how much the current solution deviates from what we expect it to be. ADMM consists of iterations of simple three steps each:

$$x^{k+1} = \text{prox}_{\lambda f}(z^k - u^k)$$
$$z^{k+1} = \text{prox}_{\lambda g}(x^{k+1} + u^k)$$
$$u^{k+1} = u^k + x^{k+1} - z^{k+1}$$
where $k$ is the iteration counter, $\text{prox}_{\lambda f}(y)$ is the proximal operator of $f(x)$ and $\text{prox}_{\lambda g}(y)$ is the proximal operator of $g(x)$.

For the tomography problem, we have the following:

- $f(x)$ is the data term, and defines the squared error of the current solution i.e. $f(x) = \|Ax - p\|_2^2$ where $x$ is the reconstructed volume and $p$ is the input projections. The proximal operator $\text{prox}_{\lambda f}(u)$ is defined by Algorithm 2.

- $g(x)$ will be any prior on the volume, for example the $\ell_1$ norm of the volume $x$ i.e. $g(x) = \|x\|_1 = \sum_i |x_i|$.

Usually we are interested in solving a slightly different problem from above i.e.

$$\min_x f(x) + g(Dx)$$

We can solve this problem using one of these algorithms:

### 3.2.1 Alternating Direction Method of Multipliers (ADMM)

Standard ADMM writes the problem as a constrained minimization by introducing a new variable $z = Dx$:

$$\min f(x) + g(z)$$

s.t. $Dx = z$

and then forms the scaled augmented Lagrangian

$$L(x, z, u) = f(x) + g(z) + \frac{\rho}{2}\|Dx - z + u\|^2$$
and then alternates minimization of the Lagrangian with updating the dual variable:

\[
x^{k+1} = \arg\min_x f(x) + \frac{\rho}{2} \| Dx - z^k + u^k \|^2 \\
z^{k+1} = \arg\min_z g(z) + \frac{\rho}{2} \| Dx^{k+1} - z + u^k \|^2 \\
u^{k+1} = u^k + D x^{k+1} - z^{k+1}
\]

Note that when \( D = I \) this becomes the usual ADMM iterations by expressing the \( x \) and \( z \) steps in the form of proximal operators:

\[
x^{k+1} = \text{prox}_{\rho f}(z^k - u^k) \\
z^{k+1} = \text{prox}_{\rho g}(x^{k+1} + u^k) \\
u^{k+1} = u^k + x^{k+1} - z^{k+1}
\]

### 3.2.2 Linearized ADMM

The first algorithm is a variation on ADMM, called Linearized ADMM [24] (also known as Split Uzawa or Inexact Split Uzawa [42, 49] or Proximal ADMM [50, 51, 52, 53]), and is defined by these three steps:

\[
x^{k+1} = \text{prox}_{\mu f}(x^k - \rho \mu D^T (D x^k - z^k + u^k)) \\
z^{k+1} = \text{prox}_{\rho g}(D x^{k+1} + u^k) \\
u^{k+1} = u^k + D x^{k+1} - z^{k+1}
\]

where the only changes are the addition of a new parameter \( \mu \) (typically \( \leq 1/4\rho \)) [24] and the modifications to the inputs of the proximal operators by incorporating the finite difference matrix \( D \).

Note that now the variables have different dimensionalities: \( x \in \mathbb{R}^n \) is the volume
of voxels, while \( z, u \in \mathbb{R}^{3n \times n} \) represent “difference volumes” (outputs of the difference operator \( D \)).

We can derive these iterations as follows: start with the normal ADMM for the minimization problem and add a special proximal term in the Augmented Lagriangian to cancel out the quadratic term containing the matrix \( D \). The problem is:

\[
\begin{align*}
\min & \quad f(x) + g(z) \\
\text{s.t.} & \quad Dx = z
\end{align*}
\]

and the normal scaled ADMM iterates the following

\[
\begin{align*}
x^{k+1} &= \arg\min_x f(x) + \frac{\rho}{2} \|Dx - z^k + u^k\|^2 \\
z^{k+1} &= \arg\min_z g(z) + \frac{\rho}{2} \|Dx^{k+1} - z + u^k\|^2 \\
u^{k+1} &= u^k + Dx^{k+1} - z^{k+1}
\end{align*}
\]

The problem with the \( x \) step is that it contains the quadratic term \( \|Dx\|^2 = x^T D^T D x \) in the minimization makes it hard to minimize since it’s not straightforward. We can cancel out that term by adding the following proximal term that makes it strongly convex and keeps the solution close to the previous iteration

\[
\frac{1}{2} \|x - x^k\|^2_S = \frac{1}{2} (x - x^k)^T S (x - x^k)
\]

to the objective function where the special matrix \( S \) is

\[
S = \frac{1}{\mu} I - \rho D^T D
\]
and this gives the modified $x$ step

\[
x^{k+1} = \arg\min_x f(x) + \frac{\rho}{2} \|Dx - z^k + u^k\|^2 + \frac{1}{2} \|x - x^k\|^2_S
\]

\[
= \arg\min_x f(x) + \frac{\rho}{2} \|Dx\|^2 - \rho \langle Dx, z^k - u^k \rangle + \frac{1}{2} (x - x^k)^T S (x - x^k)
\]

\[
= \arg\min_x f(x) + \frac{\rho}{2} \|Dx\|^2 - \rho \langle x, D^T (z^k - u^k) \rangle + \frac{1}{2} \|x\|^2 - \frac{\rho}{2} \|Dx\|^2 - \langle x, Sx^k \rangle
\]

\[
= \arg\min_x f(x) - \rho \langle x, D^T (z^k - u^k) \rangle + \frac{1}{2\mu} \|x\|^2 - \langle x, Sx^k \rangle
\]

\[
= \arg\min_x f(x) - \rho \langle x, D^T (z^k - u^k) \rangle + \frac{1}{2\mu} \|x - \mu \rho D^T (z^k - u^k) - \mu Sx^k\|^2
\]

\[
= \arg\min_x f(x) + \frac{1}{2\mu} \|x - x^k + \mu \rho D^T (z^k - u^k) - \mu Sx^k\|^2
\]

which is simply the proximal operator of $f(x)$ with input $x^k + \mu \rho (z^k - u^k - D^T D x^k)$

i.e. the iterations now become

\[
x^{k+1} = \text{prox}_{\mu f} (x^k + \mu \rho D^T (z^k - u^k - D x^k))
\]

\[
z^{k+1} = \text{prox}_{\rho^{-1} g} (D x^{k+1} + u^k)
\]

\[
u^{k+1} = u^k + D x^{k+1} - z^{k+1}
\]

where $\rho$ is the ADMM parameter and $\mu \in (0, \frac{1}{\rho \|D\|^2})$ which is what we have before for the Linearized ADMM.

The trick of the Proximal ADMM (adding this proximal term to the augmented Lagrangian) is more general, and can be used to simplify ADMM in general so that instead of having an inner loop for a minimization step that involves a linear operator/matrix (e.g. using CP algorithm or another variable splitting) it can be solved directly using the proximal operator of the function. A similar trick (of adding a proximity $\ell_2$ term to the subproblems) is actually used in the PDHG [42, 49] and the
3.2.3 Chambolle-Pock Primal-Dual Algorithm

The second algorithm is by using the Primal-Dual Algorithm of Chambolle and Pock [40]. It is also used to solve problems of the form

$$\min_x f(x) + g(Dx)$$

This is quite similar to the ADMM, and is supposed to be faster. It consists of these three steps in every iteration:

$$\begin{align*}
\bar{x}^0 &= x^0 \\
\bar{z}^{k+1} &= \text{prox}_{\mu g^*} (z^k + \mu D\bar{x}^k) \\
x^{k+1} &= \text{prox}_{\lambda f} (x^k - \lambda D^T \bar{z}^{k+1}) \\
x^{k+1} &= x^{k+1} + \theta (x^{k+1} - x^k)
\end{align*}$$

with parameters $\theta \in [0, 1]$ and $\mu$ and $\lambda$ such that $2\mu \lambda < 1$. 

CP algorithm [40].
Chapter 4

ProxiSART Framework

The overall problem we are interested in solving is a regularized data fitting problem. In particular, the objective function we are interested in minimizing is

\[ f(x) + g(Kx) \]

where \( f(\cdot) \) is a data fitting term that measures how much the solution fits the data, \( g(\cdot) \) is a regularization term that imposes constraints on acceptable solutions through multiplication by an arbitrary matrix \( K \), and usually includes another parameter \( \sigma \) that trades off the data and the regularization term i.e. it takes the form

\[ g(Kx) = \sigma \tilde{g}(Kx). \]

For example, in our case,

\[ f(x) = \|Ax - p\|_2^2 \]

which is a measure of how well the reconstruction fits the given projection images.

For the regularization part, we can have several functions. For example:

- the simplest is to regularize the \( \ell_2 \) norm of the volume i.e.

\[ g(Kx) = \sigma \tilde{g}(Kx) = \sigma \|x\|_2^2 \text{ where } K = I. \]
Another example is the $\ell_1$ norm if we have reason to believe that the volume should have few nonzero voxels i.e.

$$g(Kx) = \sigma \|x\|_1 = \sum_i |x_i|.$$  

One more example is the Anisotropic Total Variation \cite{36, 54, 55, 21}, which intuitively is the $\ell_1$ norm of the gradient of the volume i.e.

$$g(Kx) = \sigma \|\nabla x\|_1$$

where in this case the matrix $K \in \mathbb{R}^{3n \times n}$ is the matrix of first-order forward derivatives.

In order to solve this problem, we will use proximal algorithms, namely the first-order primal-dual algorithm proposed by Chambolle and Pock (henceforth referred to as the CP algorithm) \cite{40} summarized in Algorithm \ref{algorithm:cp}. For the algorithm to work, we need two proximal operators:

- The proximal operator for the first function

$$\text{prox}_{\lambda f}(u) = \arg\min_x f(x) + \frac{1}{2\lambda} \|x - u\|_2^2.$$  

- The proximal operator for the convex conjugate \cite{56} function $g^*(\cdot)$ of $g(\cdot)$ defined as

$$\text{prox}_{\mu g^*}(u) = \arg\min_x g^*(x) + \frac{1}{2\mu} \|x - u\|_2^2$$

where $g^*(\cdot)$ is defined as

$$g^*(u) = \sup_x u^T x - g(x).$$
Computing this conjugate proximal operator actually is quite easy if we know the proximal operator for $g(\cdot)$, since using Moreau’s decomposition \[24\] we have

$$\text{prox}_{\mu g^*}(u) = u - \mu \text{prox}_{\frac{1}{\mu} g}(\frac{u}{\mu}).$$ \hspace{1cm} (4.1)

It can also be shown that the proximal operator for $g^*(\cdot)$ is related to the proximal operator for $\tilde{g}^*(\cdot)$ by

$$\text{prox}_{\mu g^*}(u) = \sigma \text{prox}_{(\mu/\sigma)\tilde{g}^*}(\frac{u}{\sigma}).$$ \hspace{1cm} (4.2)

so the $z-$step of Algorithm \[3\] can be written as

$$z^{(t+1)} = \sigma \text{prox}_{(\mu/\sigma)\tilde{g}^*}\left(\frac{z^{(t)} + \mu K\tilde{x}^{(t)}}{\sigma}\right).$$

Using different regularization functions $g(\cdot)$ and matrices $K$, we can plug in different priors based on our prior information of how the reconstructed volume should look like.

### 4.1 Data Fidelity Operator

We want to find the proximal operator of the tomography problem

$$\text{prox}_{\lambda f}(u) = \arg\min_x \|p - Ax\|^2_2 + \frac{1}{2\lambda} \|x - u\|^2_2 = \arg\min_x 2\lambda \|p - Ax\|^2_2 + \|x - u\|^2_2$$

i.e. given a reconstruction $u$ find a solution $x^*$ that is close to $u$ and minimizes the reconstruction squared norm

$$\text{minimize} \hspace{1cm} 2\lambda \|p - Ax\|^2_2 + \|x - u\|^2_2$$
We can transform this problem into a least-norm problem by introducing new variables as follows. Let \( y = \sqrt{2\lambda}(p - Ax) \) and \( z = x - u \). The problem becomes

\[
\begin{align*}
\text{minimize} & \quad \|y\|^2 + \|z\|^2 \\
\text{subject to} & \quad y + \sqrt{2\lambda}Az = \sqrt{2\lambda}(p - Au)
\end{align*}
\]

or equivalently

\[
\begin{align*}
\text{minimize} & \quad \|y\|^2 + \|z\|^2 \\
\text{subject to} & \quad \begin{bmatrix} I & \sqrt{2\lambda}A \end{bmatrix} \begin{bmatrix} y \\ z \end{bmatrix} = \sqrt{2\lambda}(p - Au)
\end{align*}
\]

which can be written in the form solved by SART (since \( \tilde{A} \) is now a fat matrix):

\[
\begin{align*}
\text{minimize} & \quad \|\tilde{x}\|^2 \\
\text{subject to} & \quad \tilde{A}\tilde{x} = \tilde{p}
\end{align*}
\]

where \( \tilde{x} \in \mathbb{R}^{m+n} \)

\[
\tilde{x} = \begin{bmatrix} y \\ z \end{bmatrix}
\]

and \( \tilde{A} \in \mathbb{R}^{m \times m+n} \)

\[
\tilde{A} = \begin{bmatrix} I & \sqrt{2\lambda}A \end{bmatrix}
\]

and \( \tilde{p} \in \mathbb{R}^m \)

\[
\tilde{p} = \sqrt{2\lambda}(p - Au).
\]

This system now can be solved using SART (or any alternative of ART) to find iterates \((y^{(t)}, z^{(t)})\) of \((y, z)\) until convergence. The solution will be the least-squared solution closest to the input \(u\), and given optimal solution \(y^*\) and \(z^*\) we can recover
the optimal $x^*$ simply by $x^* = z^* + u$ or can do the iterations directly with $(y, x)$. So in principle, any ART solver “should” be easily changed to a proximal operator for the tomography reconstruction by simple manipulations of the projection matrix and the input projections.

### 4.1.1 SART Proximal Operator

The formulas for the modified SART can be easily carried over to the system $\tilde{A}\tilde{x} = \tilde{p}$ by just working with the respective updated variables:

$$
\tilde{x}_j^{(0)} = 0,
$$

$$
\tilde{x}_j^{(t+1)} = \tilde{x}_j^{(t)} + \alpha \frac{\tilde{p}_i - \sum_{k} \tilde{a}_{ik}\tilde{x}_k^{(t)} }{ \sum_{k} \tilde{a}_{ik} } \delta_{ij},
$$

which can be expanded in terms of $y$, $z$, $A$, and $\tilde{p}$:

$$
y_j^{(t+1)} = y_j^{(t)} + \alpha \frac{\tilde{p}_i - \sqrt{2\lambda} \sum_k a_{ik} z_k^{(t)} - y_j^{(t)} }{ \sqrt{2\lambda} \sum_k a_{ik} + 1 } \delta_{ij},
$$

$$
z_j^{(t+1)} = z_j^{(t)} + \alpha \frac{\sum_{i} \tilde{p}_i - \sqrt{2\lambda} \sum_k a_{ik} z_k^{(t)} - y_j^{(t)} }{ \sqrt{2\lambda} \sum_k a_{ik} + 1 } \sqrt{2\lambda} \delta_{ij},
$$

where $\delta_{ij} = 1$ when $i = j$ and zero otherwise. A few remarks:

- Only one $y_j$ is updated for every row update. I.e. when looking at projection $i$ (row $i$ in $\tilde{A}$) we only update $y_i$ and the rest get an update of zero. That is why we have the $\delta_{ij}$ term, which arises from the identity matrix $I$ in $\tilde{A}$.

- The index $j$ for $y$ runs from 1 to $m$.

- The index $j$ for $z$ runs from 1 to $n$, which is equivalent to $m + 1$ to $m + n$ on $\tilde{A}$ i.e. runs over the last $n$ columns of $\tilde{A}$ (the $n$ columns of $A$).

It is better to manipulate them to have expressions in terms of the original variable $x$, ...
the new variable $y$, and the actual inputs: $A$ the forward projection operator matrix, $p$ the projection measurements, and $u$ the input estimate or prior.

The initialization for $x$ and $y$ is:

$$\tilde{x} = \begin{bmatrix} y \\ z \end{bmatrix} = 0,$$

i.e.

$$y = 0 \quad x = u.$$ 

By manipulating the update formulas for $y$ and $z$ and using the fact that $z = x - u$ we can obtain formulas for for $y$ and $x$:

$$y_{j(t+1)} = y_{j(t)} + \alpha \sum_{i \in S} \tilde{p}_i - \sqrt{2\lambda} \sum_k a_{ik} x_{k(t)} + \frac{\sqrt{2\lambda} \sum_k a_{ik} u_k - y_{i(t)}}{\sqrt{2\lambda} \sum_k a_{ik} + 1} \delta_{ij},$$

$$x_{j(t+1)} = x_{j(t)} + \sum_{i \in S} \frac{\tilde{p}_i - \sqrt{2\lambda} \sum_k a_{ik} x_{k(t)} + \sqrt{2\lambda} \sum_k a_{ik} u_k - y_{i(t)}}{\sqrt{2\lambda} \sum_k a_{ik} + 1} \sqrt{2\lambda} a_{ij} \delta_{ij}.$$ 

Using the fact that $\tilde{p} = \sqrt{2\lambda} (p - Au)$ which gives

$$\tilde{p}_i = \sqrt{2\lambda} p_i - \sqrt{2\lambda} \sum_k a_{ik} u_k,$$

we can further simplify the update equations to

$$y_{j(t+1)} = y_{j(t)} + \alpha \sum_{i \in S} \frac{\sqrt{2\lambda} p_k - \sqrt{2\lambda} \sum_k a_{ik} x_{k(t)} - y_{i(t)}}{\sqrt{2\lambda} \sum_k a_{ik} + 1} \delta_{ij},$$

$$x_{j(t+1)} = x_{j(t)} + \frac{\sum_{i \in S} \sqrt{2\lambda} p_k - \sqrt{2\lambda} \sum_k a_{ik} x_{k(t)} - y_{i(t)}}{\sqrt{2\lambda} \sum_k a_{ik} + 1} \sqrt{2\lambda} a_{ij} \delta_{ij}.$$
Putting it together and simplifying a bit, we can write it as in Algorithm 2.

**Algorithm 2** SART Proximal Operator

**Require:** $A \in \mathbb{R}^{m \times n}$, $u \in \mathbb{R}^n$, $\lambda \in \mathbb{R}$, $\alpha \in \mathbb{R}$, $p \in \mathbb{R}^m$

1. $p = \sqrt{2\lambda p}$
2. $A = \sqrt{2\lambda A}$
3. Initialize
   
   \[
   y^{(0)} = 0 \\
   x^{(0)} = u
   \]

4. for all $t = 1 \ldots T$ do
5.   for all projections $S \in S_1 \ldots S_N$ do
6.     \[
   y_j^{(t+1)} = y_j^{(t)} + \alpha c_j^{(t)} \quad \text{for } j \in S
   \]
7.     \[
   x_j^{(t+1)} = x_j^{(t)} + \alpha \frac{\sum_{i \in S} c_i^{(t)} a_{ij}}{\sum_{i \in S} a_{ij}} \quad \text{for } j = 1 \ldots n
   \]
   where
8.     \[
   c_i^{(t)} = \frac{p_l - \sum_k a_{ik} x_k^{(t)} - y_i^{(t)}}{\sum_k a_{ik} + 1}
   \]
9.   end for
10. end for
11. return volume reconstruction $x \in \mathbb{R}^n$

### 4.1.2 Discussion

So the changes from a standard SART:

1. The volume $x$ is initialized with $x = u$ using the input volume. A volume of say 256 $\times$ 256 $\times$ 256 and 4 bytes per voxel, takes $2^{26} = 64$ MB of storage, and a volume of 1024 $\times$ 1024 $\times$ 1024 takes $2^{32} = 4$ GB.

2. Keeping around a new variable $y \in \mathbb{R}^m$ where $m$ is the number of line projections (total number of pixels in the projection images). Assuming 360 projections of 3 MP images (2000 $\times$ 1500 images) and 4 bytes each, this adds up about 4 GB of RAM.
3. For our data, if everything goes in RAM, we will need to keep one version of
the volume $x$ and two versions of the projections $p$ and $y$, which would take ~
12 GB.

4. Updating both $y$ and $x$ at each iteration:

(a) The update for $y$ is very fast because only one index $y_j$ is updated for every
projection pixel $i = j$.

(b) The update for $x$ is very similar to standard SART with the exception of
the term $y_i^{(t)}$ in the formula for $c_i^{(t)}$.

### 4.2 Regularization Operators

#### 4.2.1 L1 norm

The proximal operator for the $\ell_1$ norm is the *soft thresholding* or *shrinkage* function

$$\text{prox}_{\lambda g}(u) = (u - \lambda)_+ - (-u - \lambda)_+$$

where $(u)_+ = \max(u, 0)$ is the positive part of $u$, and can also be written as

$$(\text{prox}_{\lambda g}(u))_i = \begin{cases} 
  u_i - \lambda & u_i \geq \lambda \\
  0 & |u_i| \leq \lambda \\
  u_i + \lambda & u_i \leq -\lambda
\end{cases}$$

where $(\text{prox}_{\lambda g}(u))_i$ is the $i^{th}$ element of the output.
4.2.2 Anisotropic Total Variation Prior (ATV)

A better prior is the Anisotropic Total Variation prior

\[ h(x) = \sum_{ijk} |x_{i+1,j,k} - x_{i,j,k}| + |x_{i,j+1,k} - x_{i,j,k}| + |x_{i,j,k+1} - x_{i,j,k}| \]

where \( x_{i,j,k} \) is the voxel value at position \((i, j, k)\). This can be represented as \( g(Dx) \) where \( D \in \mathbb{R}^{3n \times n} \) is the forward difference matrix (that for every voxel computes the forward difference to the three voxels to the right, bottom, and back) and \( g(w) = \|w\|_1 \) is the \( \ell_1 \) norm i.e. \( h(x) = \|Dx\|_1 \).

The two proximal operators are:

- the proximal operator related to the prior \( g(u) \):

\[
\text{prox}_{\mu g^*}(u) = u - \mu \text{prox}_{\frac{1}{\mu} g} \left( \frac{u}{\mu} \right)
\]

is the proximal operator of the convex conjugate of \( g(u) \) \[23\], and in our case, where \( g(u) = \|u\|_1 \) would be

\[
\text{prox}_{\mu g^*}(u) = u - \mu \left( \frac{u}{\mu} - \frac{1}{\mu} \right)_+ - \left( -\frac{u}{\mu} - \frac{1}{\mu} \right)_+
\]

\[
= u - (u - 1)_+ + (-u - 1)_+
\]

\[
= \begin{cases} 
1 & u > 1 \\
|u| & |u| \leq 1 \\
-1 & u < -1 
\end{cases}
\]

or we can deduce it another way since we know that the conjugate of \( g(u) \) is

\[
g^*(u) = I_{B_\infty}(u) = \begin{cases} 
0 & \text{if } \|u\|_\infty \leq 1 \\
\infty & \text{otherwise}
\end{cases}
\]
which is the indicator function of unit ball $B_\infty$ of the $\ell_\infty$ norm

$$\|u\|_\infty = \max_i |u_i|$$

and its proximal operator is the projection on its unit ball

$$\text{prox}_{\mu g^*}(u) = P_{B_\infty}(u) = \begin{cases}
1 & u > 1 \\
u & |u| \leq 1 \\
-1 & u < -1
\end{cases}$$

which is the same as before.

- and the proximal operator related to the data term $f(x)$ which is the normal SART proximal operator.

The dimensionality of the data here is also different. $x, \bar{x} \in \mathbb{R}^n$ denote volumes, while $z \in \mathbb{R}^{3n \times n}$ denote difference volumes (outputs of the difference operator $D$).

We wouldn’t need to store the matrix $D$ explicitly, rather we can compute it on the fly. For example, $Dx$ means we compute the volume whose elements are the finite forward difference in each of the three dimensions. One possible choice for $D$ can be as follows [12]: Imagine a graph where every voxel is a vertex and there are edges connecting every voxel to its six neighbors. The graph will have $n$ vertices (voxels) and $3n$ edges (3 values per vertex, one per dimension). Each edge will correspond to one finite difference calculation, and $D$ will have a row for every edge and a column for every vertex (voxels). Consider any edge $e$ connecting voxel $i$ to voxel $j$, the
corresponding values of $D$ in that row are:

$$
D_{ev} = \begin{cases} 
-1 & \text{if } v = i \\
1 & \text{if } v = j \\
0 & \text{otherwise}
\end{cases}
$$

where we are assuming that voxels at the boundary are replicated, and hence get a difference of zero.

Multiplying by $D^T$ can similarly be achieved without explicitly storing $D$. To compute $D^T z$ for some volume $z$, consider a specific vertex $v$. The value of $D^T z$ for that vertex

$$(D^T z)_v = \sum_e D_{ev} z_e$$

where $D_{ev} = \pm 1$ as defined above. In other words, if $z$ represents some difference volume (e.g. $z = Dw$ for some $w$), $(D^T z)_v$ sums up the difference values (edges) that this edge $v$ was involved in, where the coefficient is either 1 or $-1$ depending on the position of that vertex.

For example, consider a vertex $v = (i, j, k)$ of a volume $x$. This vertex (voxel) will be involved in six differences (i.e. six edges), and will contribute six entries in the output $z = Dx$ (two for each dimension):

$$z_v = \begin{bmatrix} 
x(i+1,j,k) - x(i,j,k) 
x(i,j+1,k) - x(i,j,k) 
x(i,j,k+1) - x(i,j,k) 
x(i,j,k) - x(i-1,j,k) 
x(i,j,k) - x(i,j-1,k) 
x(i,j,k) - x(i,j,k-1) 
\end{bmatrix},$$

when computing $D^T Dx$, the entry corresponding to vertex $v$ will sum up these six
entries with the corresponding coefficient ±1:

\[
\begin{bmatrix}
-1 \\
-1 \\
-1 \\
1 \\
1
\end{bmatrix} = \sum_m z_v[m]
\]

\[
= 6x(i, j, k) - x(i - 1, j, k) - x(i, j - 1, k) - x(i, j, k - 1) - x(i + 1, j, k) - x(i, j + 1, k) - x(i, j, k + 1)
\]

### 4.2.3 Isotropic Total Variation Prior (ITV)

The *Isotropic Total Variation* function doesn’t separate the individual gradient components, but rather sums the magnitudes of the individual gradients at each voxel

\[
h(x) = \sum_{ijk} \sqrt{|x_{i+1,j,k} - x_{i,j,k}|^2 + |x_{i,j+1,k} - x_{i,j,k}|^2 + |x_{i,j,k+1} - x_{i,j,k}|^2}
\]

Using the same matrix \(D \in \mathbb{R}^{3n \times n}\) as defined above, and defining a new matrix \(E \in \mathbb{R}^{3n \times n}\) that denotes the positions of the forward differences [42] such that

\[
E_{ev} = \begin{cases} 
1 & \text{if } D_{ev} = -1 \\
0 & \text{otherwise}
\end{cases}
\]

we can define the function \(h(x)\) as a norm \(\|w\|_E\) for \(w = Dx \in \mathbb{R}^{3n}\) defined as

\[
\|w\|_E = \|E^T w^2\|_1 = \sum_v \|w^v\|_2
\]

where the square root and square functions are component-wise, and \(w^v\) is the gradient
at voxel $v = (i, j, k)$ i.e.

$$w^v = \begin{bmatrix}
  x(i + 1, j, k) - x(i, j, k) \\
  x(i, j + 1, k) - x(i, j, k) \\
  x(i, j, k + 1) - x(i, j, k)
\end{bmatrix}.$$ 

The dual norm $\|w\|_{E^*}$ is by definition

$$\|w\|_{E^*} = \max_{\|y\|_{E} \leq 1} y^T w$$

and can be shown to be \cite{42} equal to

$$\|w\|_{E^*} = \max_v \|w^v\|_2 = \|\sqrt{E^T w^2}\|_{\infty}$$

in other words it is the maximum gradient magnitude of $w = Dx$ in $x$.

Now we can express the TV prior $h(x)$ in terms of the $\|w\|_{E}$ norm as

$$h(x) = \|Dx\|_{E} = g(Dx)$$

where $g(w) = \|w\|_{E}$.

To be used with ADMM or the Primal-Dual algorithm, we need to find the proximal operator of the function $g(w) = \|w\|_{E}$. Using relations between the norm and its dual and the properties of the proximal operators \cite{23, 24, 42}, we can show that the proximal operator of the function $g(w)$ is

$$\text{prox}_{\lambda g}(u) = u - \lambda P_{B^*}(\frac{u}{\lambda})$$

where $P_{B^*}(u)$ is the projection on the unit ball of the dual norm i.e. the set $B^* = \{w \mid \|w\|_{E^*} \leq 1\}$, and in fact it is related proximal operator of the conjugate function.
$g \ast (u)$ of $g(u)$ which is the indicator function of the dual norm unit ball

$$g \ast (u) = \begin{cases} 0 & \|u\|_{E^*} \leq 1 \\ \infty & \text{otherwise} \end{cases}$$

and has a proximal operator

$$\text{prox}_{\lambda g^*}(u) = P_{B^*}(u)$$

This projection function can be shown to be \cite{42, 39}

$$P_{B^*}(u) = \arg \min_{p \in B^*} \|p - u\|^2 = \frac{u}{E \max \left( \sqrt{E^T u^2}, 1 \right)}$$

where the division and max operations are again component-wise operations. It can also be written as

$$(P_{B^*}(u))_e = \frac{u_e}{\max (\|w_v\|_2, 1)}$$

where $e$ is the entry in $u$ (an edge) and $v$ is the corresponding voxel (vertex) where $e$ originates from. In other words, the entries of $u$ are scaled by the magnitude of the gradient at each voxel. For $u = Dx$, we can also view it as a scaling of the gradient by its magnitude if the magnitude exceeds unity and leaving it unchanged otherwise i.e.

$$P_{B^*}(Dx) = \begin{cases} \frac{\nabla x}{|\nabla x|} & \text{if } |\nabla x| \geq 1 \\ \nabla x & \text{otherwise} \end{cases}$$

Now given the proximal operator of $g(u)$ defined above, we can plug it into either Linearized ADMM to regularize with the isotropic total variation. For the Primal-Dual where the proximal operator of the conjugate function is required, we can use $\text{prox}_{\lambda g^*}(u) = P_{B^*}(u)$ defined above directly instead.
4.2.4 Sum of Absolute Differences (SAD)

This prior is defined as:

\[ h(x) = \sum_{ijk} \sum_{x_n \in N(x_{i,j,k})} |x_n - x_{i,j,k}| \]  

(4.3)

where \( N(x_{i,j,k}) \) is the \( 3 \times 3 \) neighborhood around voxel \( x_{i,j,k} \) (excluding voxel \( x_{i,j,k} \) itself). It can be seen as an extension to the ATV prior, just with a different matrix \( D \) where more edges are considered for every voxel instead of just three.

4.3 Solvers

In order to solve this problem, we will use proximal algorithms, namely the first-order primal-dual algorithm proposed by Chambolle and Pock (henceforth referred to as the CP algorithm) [40] summarized in Algorithm 3. For the algorithm to work, we need two proximal operators:

- The proximal operator for the first function

\[ \text{prox}_{\lambda f}(u) = \arg\min_x f(x) + \frac{1}{2\lambda} \|x - u\|_2^2. \]

- The proximal operator for the convex conjugate [56] function \( g^*(\cdot) \) of \( g(\cdot) \) defined as

\[ \text{prox}_{\mu g^*}(u) = \arg\min_x g^*(x) + \frac{1}{2\mu} \|x - u\|_2^2 \]

where \( g^*(\cdot) \) is defined as

\[ g^*(u) = \sup_x u^T x - g(x). \]

Computing this conjugate proximal operator actually is quite easy if we know
the proximal operator for \( g(\cdot) \), since using Moreau’s decomposition \cite{24} we have

\[
\text{prox}_{\mu g^*}(u) = u - \mu \text{prox}_{\frac{1}{\mu} g}(\frac{u}{\mu}).
\]  

(4.4)

It can also be shown that the proximal operator for \( g^*(\cdot) \) is related to the proximal operator for \( \tilde{g}^*(\cdot) \) by

\[
\text{prox}_{\mu g^*}(u) = \sigma \text{prox}_{(\mu/\sigma)\tilde{g}^*}(\frac{u}{\sigma}),
\]  

(4.5)

so the \( z \)-step of Algorithm \[3\] can be written as

\[
z^{(t+1)} = \sigma \text{prox}_{(\mu/\sigma)\tilde{g}^*}\left(\frac{z^{(t)} + \mu K \bar{x}^{(t)}}{\sigma}\right).
\]

Using different regularization functions \( g(\cdot) \) and matrices \( K \), we can plug in different priors based on our prior information of how the reconstructed volume should look like.

\begin{algorithm}
\SetAlgoLined
\textbf{Require:} \( K \in \mathbb{R}^{d \times n} \), \( \theta \in [0, 1] \), \( \lambda, \mu \) such that \( \mu \lambda \|K\|^2 < 1 \), \( \sigma \in \mathbb{R} \), initial values \( x^{(0)} \in \mathbb{R}^n \) and \( z^{(0)} \in \mathbb{R}^d \)
\textbf{1:} Initialize
\[ \bar{x}^{(0)} = x^{(0)} \]
\textbf{2:} for all \( t = 1 \ldots T \) do
\[ z^{(t+1)} = \text{prox}_{\mu g^*}\left( z^{(t)} + \mu K \bar{x}^{(t)} \right) \]
\[ x^{(t+1)} = \text{prox}_{\lambda f}\left( x^{(t)} - \lambda K^T z^{(t+1)} \right) \]
\[ \bar{x}^{(t+1)} = x^{(t+1)} + \theta(x^{(t+1)} - x^{(t)}) \]
\textbf{3:} end for
\textbf{return} \( x^{(T)} \in \mathbb{R}^n \)
\end{algorithm}
4.4 Framework Implementation Detail

We use our SART proximal operator detailed in Algorithm 2 for solving $\text{prox}_f(\cdot)$ required in the CP algorithm. The implementation of $\text{prox}_{g^\ast}(\cdot)$ depends on the choice of prior. We experiment with the following priors:

- Anisotropic TV (ATV)

This prior is defined as:

$$h(x) = \sum_{ijk} |x_{i+1,j,k} - x_{i,j,k}| + |x_{i,j+1,k} - x_{i,j,k}| + |x_{i,j,k+1} - x_{i,j,k}|$$  \hspace{1cm} (4.6)

where $x_{i,j,k}$ is the voxel value at position $(i,j,k)$. This can be represented as $\tilde{g}(Kx)$ where $g(\cdot) = \| \cdot \|_1$ is the $\ell_1$ norm and $K = D \in \mathbb{R}^{3n \times n}$ is the forward difference matrix defined as

$$D_{ev} = \begin{cases} 
-1 & \text{if } v = i \\
1 & \text{if } v = j \\
0 & \text{otherwise} 
\end{cases}$$  \hspace{1cm} (4.7)

where we consider a graph defined on the volume such that we have $n$ vertices (voxels) and $3n$ edges (connecting each voxel to its six neighbors, replicating voxels at the edge). The matrix $D$ is a representation of this graph, where each row corresponds to one edge, and each column to one vertex. The value at row $e$ and column $v$ is given by Eq. 4.7 above. The proximal operator of $\tilde{g}^\ast(\cdot)$ can
be shown to be \[ \varpropto_{\mu_{\hat{g}}}(u) = P_{B_{\infty}}(u) = \begin{cases} 
1 & u > 1 \\
|u| & |u| \leq 1 \\
-1 & u < -1 \end{cases} \] (4.8) where the operations are component-wise, and is equivalent to the projection on the unit ball \( B_{\infty} \) of the \( \ell_{\infty} \) norm. Note that we do not need to store the matrix \( D \), and multiplication by \( D \) (computing the gradient) or by \( D^T \) (computing the divergence) can be efficiently computed on-the-fly \[42\].

- Isotropic TV (ITV)

This prior is defined as:

\[
h(x) = \sum_{ijk} \sqrt{|x_{i+1,j,k} - x_{i,j,k}|^2 + |x_{i,j+1,k} - x_{i,j,k}|^2 + |x_{i,j,k+1} - x_{i,j,k}|^2}
\]

where it sums the magnitude of the gradient at each voxel. Using the same matrix \( D \) defined in Eq. 4.7 and defining a new matrix \( E \in \mathbb{R}^{3n \times n} \) that denotes the positions of the forward differences \[42\] such that

\[
E_{ev} = \begin{cases} 
1 \quad \text{if } D_{ev} = -1 \\
0 \quad \text{otherwise}
\end{cases}
\]

we can define the function \( h(x) \) as a norm \( \|u\|_E \) for \( u = Dx \in \mathbb{R}^{3n} \) defined as

\[
\|w\|_E = \|\sqrt{E^T}w\|_1 = \sum_v \|w^v\|_2
\]

where the square root and square functions are component-wise, and \( w^v \) is the
gradient at voxel \( v = (i, j, k) \) i.e.

\[
w^v = \begin{bmatrix}
  x_{i+1,j,k} - x_{i,j,k} \\
  x_{i,j+1,k} - x_{i,j,k} \\
  x_{i,j,k+1} - x_{i,j,k}
\end{bmatrix}.
\]

Now we can express the ITV prior \( h(x) \) in terms of the \( \|u\|_E \) norm as

\[
h(x) = \|Dx\|_E = g(Dx) \quad \text{where} \quad \tilde{g}(u) = \|u\|_E.
\]

The proximal operator for \( g^*(\cdot) \) can be shown to be \([42]\)

\[
\text{prox}_{\mu\tilde{g}^*}(u) = P_{B^*}(u) = \frac{u}{E \max\left(\sqrt{E^T u^2}, 1\right)} \quad (4.9)
\]

which is the projection on the unit ball \( B^* \) of the dual norm \( \|u\|_{E^*} \), and where the division and max operations are performed component-wise. Intuitively, this scales each voxel by the magnitude of its gradient if it’s greater than unity. Similar to ATV, we don’t need to explicitly store the matrix \( E \), and multiplication by \( E \) can be computed efficiently on-the-fly.

- Sum of Absolute Differences (SAD)

This prior is defined as:

\[
h(x) = \sum_{ijk} \sum_{x_n \in N(x_{i,j,k})} |x_n - x_{i,j,k}| \quad (4.10)
\]

where \( N(x_{i,j,k}) \) is the \( 3 \times 3 \) neighborhood around voxel \( x_{i,j,k} \) (excluding voxel \( x_{i,j,k} \) itself). It can be seen as an extension to the ATV prior, just with a different matrix \( D \) where more edges are considered for every voxel instead of just three. Hence its proximal operator is similar to Eq. [4.8] It has been shown
to produce excellent results in stochastic tomography reconstruction.
Chapter 5

Experiment

5.1 Experiment design

We run two kinds of experiments: (a) on simulated data using the Shepp-Logan phantom [8]; and (b) on real data using the standard RabbitCT benchmark [58] and the Lynx dataset. All experiments are executed on a machine with two Intel Xeon X5650 processors (24 cores overall) and 48 GB of RAM. The code is implemented in C++, and our implementation of SART uses the Kaiser-Bessel kernels [59] (with a radius of 2 voxels) for more accurate and smooth interpolation in the forward and back projection steps. We also employ the slice-by-slice mechanism to speed up the (back)projection steps [59]. We evaluate our results using two metrics: (a) Signal-to-noise Ratio (SNR); and (b) Structural Similarity Index (SSIM) [60]. The first metric is a standard one used for measuring the quality of volume reconstruction compared to the ground truth volume, while the second gives more perceptually-related reconstruction measure.

5.1.1 Comparison Groups

We run different versions of our algorithms with different priors, namely:

- Our implementation of plain SART with no priors as in Alg. [1] (PlainSART).
- ProxiSART with Anisotropic Total Variation using CP (PSART-ATV)
- ProxiSART with Isotropic Total Variation using CP (PSART-ITV).
- ProxiSART with Sum of Absolute Differences using CP (PSART-SAD).

We compare results from our framework to state-of-the-art algorithms and comparable implementations in RTK, namely:

- Cone Beam Filtered Back Projection (RTK-FDK) [9].
- Plain SART with no priors (RTK-SART).
- ADMM with ATV prior (RTK-ADMM) using Conjugate Gradient (CG) [37].

For choosing the hyper parameters in all the algorithms, we experiment with a range of combinations and pick the one with the best performance (SNR).

## 5.2 Simulated Dataset

We run experiments on a $128 \times 128 \times 128$ 3D Shepp-Logan volume with voxel size of $1 \times 1 \times 1$ mm. The distance between the cone beam source and the detector matrix is 955 mm, while that between the source and the object isocenter is 500 mm. The detector panel has $512 \times 512$ pixels with $1 \times 1$ mm size. We focus on cases where noise might prevent accurate reconstruction. Therefore, we compare reconstruction using different numbers of projection images; namely 30, 45, 60, and 90 images equally distributed across the $360^\circ$ angular range. We also compare injecting zero-mean additive Gaussian noise with different standard deviations; namely 0.05, 0.2, 0.5, and 0.75. The projections were generated using RTK [45]. We set the number of SART and CG iterations at 2 in every step for our algorithms and for RTK-ADMM, respectively. The relaxation coefficient $\alpha$ in SART is set to 0.5.
Figure 5.1: Effect of the number of projections showing SNR (left) and SSIM (right) as a function of the number of projections used using Shepp-Logan phantom.

Figure 5.2: Effect of noise showing SNR (left) and SSIM (right) as a function of the amount of injected noise using Shepp-Logan phantom.

Fig. 5.1 shows the SNR and SSIM for the different reconstruction algorithms when varying the number of projections. Fig. 5.4 shows a sample slice from the reconstructed volume for the different algorithms and different numbers of projections. Similarly, Fig. 5.2 shows the SNR and SSIM for the different algorithms with different amounts of additive Gaussian noise, while Fig. 5.5 shows a sample slice from the respective reconstructions (we omit results for PSART-ATV because they are almost identical to PSART-ITV). To assess the speed of convergence, Fig. 5.3 shows the SNR as a function of the number of iterations in the reconstruction, with 60 projections and $\sigma = 0.75$ noise.

5.3 Real Datasets

We run another round of experiments on a real datasets, namely: (a) the RabbitCT [58]; and (b) the Lynx Dataset. The RabbitCT is a CT scan of a rabbit used for
Figure 5.3: Speed of convergence: SNR as a function of the number of iterations for the Shepp-Logan phantom with 60 projections and $\sigma = 0.75$ Gaussian noise.

Figure 5.4: A sample slice with different number of projections for different algorithms. From top to bottom: 90, 60, 45, and 30 projections. From left to right: RTK-FDK, RTK-SART, PlainSART, RTK-ADMM, PSART-ITV, PSART-SAD. The displayed range of values is $[0.97, 1.09]$. 
Figure 5.5: A sample slice with amounts of noise for different algorithms. From top to bottom: standard deviations of 0.05, 0.2, 0.5, and 0.75. From left to right: RTK-FDK, RTK-SART, PlainSART, RTK-ADMM, PSART-ITV, PSART-SAD. The displayed range of values is [0.97, 1.09].
benchmarking the speed of tomographic reconstruction algorithms. It contains 496 projections obtained using a C-arm system from Siemens Artis Zee acquired on a $200^\circ$ circular short-scan trajectory. We use 62 ($=496/8$) equally-distributed projections. The size of the projection image is $1248 \times 960$ pixels with isotropic resolution of 0.32 mm/pixel. The source to detector distance is 871.1088 mm and the source to isocenter is 600 mm. We used a volume of size $128 \times 128 \times 128$ voxels and 2 mm spacing.

The Lynx dataset is a CT scan of a lynx skull using an Agfa Orthoregular intensifier screen. We use 90 equally-distributed projections. The source to detector distance is 784.86 mm and the source to isocenter distance is 250 mm. The detector size is $546 \times 348$ pixels with $1 \times 1$ mm spacings. The volume size is $350 \times 260 \times 260$ voxels with $0.5 \times 0.5 \times 0.5$ mm spacing.

We use 20 iterations for both ADMM and CP (with 2 iterations for SART and CG, respectively). Fig. 5.6 and 5.7 shows sample slices from reconstruction of the RabbitCT and the Lynx datasets, respectively. We notice that using ProxiSART with ITV or SAD provides better reconstructions with less artifacts than FDK or plain SART.

5.4 Discussion

We note the following:

- Using priors (ATV, ITV, or SAD) consistently gives better accuracy and faster convergence, specially with noisy measurements or using very few projections.

- Our implementation of SART converges faster and gives lower error than RTK-SART. One reason might be the better interpolation filter (Kaiser-Bessel vs. tri-linear in RTK) we use.

- Our ProxiSART reconstructions consistently give better results than the equivalent RTK-ADMM.
Figure 5.6: Sample slices for the reconstruction of the RabbitCT dataset in the Sagittal (top), Axial (middle), and Coronal (bottom) planes. From left to right: RTK-FDK, RTK-SART, PlainSART, RTK-ADMM, PSART-ITV, PSART-SAD.

Figure 5.7: Sample slices for the reconstruction of the Lynx dataset in the Sagittal (top), Axial (middle), and Coronal (bottom) planes. From left to right: RTK-FDK, RTK-SART, PlainSART, RTK-ADMM, PSART-ITV, PSART-SAD.
• ITV and ATV priors give almost identical results, hence we include only the visualizations for ITV in Fig. 5.4 and 5.5.

• PSART-SAD produces the best results in terms of visualization quality, SNR, and convergence speed, confirming earlier results about the SAD regularizer [57].
Chapter 6

Concluding Remarks

6.1 Summary

We presented ProxiSART, a flexible proximal framework for robust 3D cone beam reconstruction. We derived the proximal operator for the data fitting sub-problem using the powerful SART algorithm. We ran experiments comparing our framework with the popular RTK open-source software toolkit, both on real and simulated datasets, using different standard and non-standard priors. We showed the robustness of our algorithms in terms of reconstruction quality and fewer iterations till convergence in the presence of noise and using fewer projections.

6.2 Future Work

Possibilities for future work include the exploration of using other more powerful priors, specifically structure tensor-based priors and non-local priors; and using the methods with other noise models than the Gaussian noise model, for example the Poisson noise model.
REFERENCES


[27] L. Grady, V. Singh, T. Kohlberger, C. Alvino, and C. Bahlmann, “Automatic segmentation of unknown objects, with application to baggage security,” in


