High-Level Algorithm Prototyping: An Example Extending the TVR-DART Algorithm

Axel Ringh[1(✉)], Xiaodong Zhuge[2], Willem Jan Palenstijn[2], Kees Joost Batenburg[2,3], and Ozan Öktem[1]

[1] Department of Mathematics, KTH Royal Institute of Technology, Stockholm, Sweden
{aringh,ozan}@kth.se
[2] Computational Imaging, Centrum Wiskunde & Informatica (CWI), Amsterdam, The Netherlands
{x.zhuge,willem.jan.palenstijn,joost.batenburg}@cwi.nl
[3] Mathematical Institute, Leiden University, Leiden, The Netherlands

Abstract. Operator Discretization Library (ODL) is an open-source Python library for prototyping reconstruction methods for inverse problems, and ASTRA is a high-performance Matlab/Python toolbox for large-scale tomographic reconstruction. The paper demonstrates the feasibility of combining ODL with ASTRA to prototype complex reconstruction methods for discrete tomography. As a case in point, we consider the total-variation regularized discrete algebraic reconstruction technique (TVR-DART). TVR-DART assumes that the object to be imaged consists of a limited number of distinct materials. The ODL/ASTRA implementation of this algorithm makes use of standardized building blocks, that can be combined in a plug-and-play manner. Thus, this implementation of TVR-DART can easily be adapted to account for application specific aspects, such as various noise statistics that come with different imaging modalities.

1 Introduction

Inverse problems refer to the task of reconstructing parameters characterizing the system under investigation from indirect observations. Such problems arise in several areas of science and engineering, and in particular for *tomographic imaging*. The idea here is to expose the object to penetrating waves or particles

A. Ringh and O. Öktem—The authors are supported by the Swedish Research Council (VR) grant 2014-5870, and the Swedish Foundation of Strategic Research (SSF) grant AM13-0049.

X. Zhuge—The author is supported by the Stichting voor de Technische Wetenschappen (STW) through a personal grant (Veni, 13610).

W.J. Palenstijn—The author is supported by the Stichting voor de Technische Wetenschappen (STW), project 13314.

K.J. Batenburg—The author is supported by the Netherlands Organization for Scientific Research (NWO), project 639.073.506.

W.G. Kropatsch et al. (Eds.): DGCI 2017, LNCS 10502, pp. 109–121, 2017.
DOI: 10.1007/978-3-319-66272-5_10

from different directions. The measured transmission (or emission) data is then used as input to a reconstruction scheme that computes an estimate of the interior structure of the object.

Computed tomography (CT) has a wide range of applications, e.g., X-ray CT [9] in medical imaging and electron tomography (ET) [8,10] in biology and material science. A key element is to model the interaction between the object and the wave/particle probe with sufficient accuracy. The resulting inverse problems are often ill-posed, for example in the sense that small errors in data get amplified. Hence, one must stabilize the reconstruction (regularization) by exploiting a priori knowledge of the unknown interior structure. Discrete tomography considers a specific type of prior knowledge, where it is assumed that the unknown object consists of a small number of different materials, each corresponding to a characteristic, approximately constant grey level in the reconstruction. A variety of reconstruction algorithms have been proposed for discrete tomography problems including primal-dual subgradient algorithms [12], network flow algorithms [3], statistical methods [2,7] and algebraic methods [4,17].

At first sight it may seem that most aspects of a reconstruction method are problem-specific. This is fortunately not the case. In fact, the general theory developed during the last three decades provides a number of generic frameworks that are adaptable to specific ill-posed inverse problems. When properly adapted, the methods derived from the general framework compare favorably with application-specific approaches. Furthermore, using general mathematical tools to address specific problems also provides new insights that may go unnoticed if one uses an entirely problem-specific approach. An example is sparsity promoting regularization, which is a general framework that outperforms, or matches, many application-specific state-of-the-art approaches for inverse problems where data are highly noisy or under-sampled.

Despite the above, most concrete implementations of reconstruction methods are tied to a specific application in the sense that minor mathematical modifications lead to a large amount of low level implementations and modifications, which require substantial dedicated algorithmic and programming efforts. Operator Discretization Library (ODL) [1] and ASTRA toolbox [14] are two open-source software libraries developed to assist fast prototyping of reconstruction algorithms. When used together, a user may implement a generic reconstruction method and use it on different tomographic real-world problems without having to re-implement all necessary parts from the bottom up. This becomes especially useful for complex reconstruction methods. *This paper demonstrates the capabilities of ODL and ASTRA on a recently proposed discrete tomography algorithm, total variation regularized discrete algebraic reconstruction technique (TVR-DART).*

2 Inverse Problems and Tomography

Mathematically, an inverse problem in imaging can be stated as the problem of reconstructing an image $f \in X$ representing the object under study from data

$g \in Y$ where

$$g = \mathcal{A}(f) + \text{“noise”}. \tag{1}$$

Here, $\mathcal{A} : X \to Y$ (forward operator) models how an image gives rise to data in the absence of noise. Moreover, X is a suitable Hilbert space of real valued functions supported on a fixed domain $\Omega \subset \mathbb{R}^n$ whose elements represent attenuation of emission values. Likewise, Y is a Hilbert space of real-valued functions that represent data and that are defined on some manifold $\mathbb{M}$ (data manifold).

In *tomographic imaging*, data can often be modeled as line integrals of the function f that describes the object along a line, i.e., data is a real-valued function on some set of lines $\mathbb{M}$ in $\mathbb{R}^n$. We can now introduce coordinates on this data manifold. A line in $\mathbb{R}^n$ can be described by a directional vector in the unit sphere S^{n-1} and a point that it passes through. This provides coordinates on $\mathbb{M}$ where a line is given by $(\omega, x) \in S^{n-1} \times \mathbb{R}^n$ with $x \in \omega^\perp$. Here, $\omega^\perp \subset \mathbb{R}^n$ is the unique plane through the origin with $\omega \in S^{n-1}$ as its normal vector. The corresponding forward operator $\mathcal{A}$ is the *ray transform*, which is expressible in the aforementioned coordinates as

$$\mathcal{A}(f)(\omega, x) := \int_{-\infty}^{\infty} f(x + t\omega)dt \quad \text{for } f \in X. \tag{2}$$

Tomographic data can then be seen as values of $\mathcal{A}(f)(\omega, x)$ for a sampling of $\omega \in S^{n-1}$ (angular sampling) and $x \in \omega^\perp$ (detector sampling). With slight abuse of terminology, one refers to these data as the “projection” of f along the line given by (ω, x).

3 Overview of ODL and ASTRA

Many reconstruction schemes can be formulated in a generic, yet adaptable, manner by stating them in an abstract coordinate-free setting using the language of functional analysis. The adaptability stems from “parametrizing” the scheme in terms of the forward operator, the data noise model, and the type of a priori information that one seeks to exploit. These can be further broken down into more *generic* components, each representing a well-defined generally applicable mathematical structure or operation. Another advantage that comes with a generic formulation is that it makes the reconstruction scheme more transparent.

These considerations form a natural blueprint for a modular software library for inverse problems where the forward operator, data noise model, and prior model are treated as independent exchangeable components. ODL is such a software library whose *design principles* are *modularity*, *abstraction*, and *compartmentalization* [1] that is freely available at http://github.com/odlgroup/odl. To realize these design principles, ODL separates *which* mathematical object or operation one seeks to represent from *how* it is implemented using concrete computational routines. Mathematical objects and operations are represented by abstract classes with abstract methods and specific implementations are represented by concrete subclasses. Hence, these abstract classes form a domain

specific language for functional analysis and corresponding subclasses couple to relevant numerical libraries. In this way one can express abstraction in a way that allows combining generic and application-specific code. The generic part, such as an optimization method, can be formulated in a coordinate-free manner using abstract classes and methods, whereas application-specific parts, such as evaluating the forward operator, are contained in specific subclasses. Hence, one can express reconstruction schemes using a clean near-mathematical syntax and involved implementation specific details are hidden in concrete subclasses.

A key part of ODL is the notion of an operator between two vector spaces. ODL offers operator calculus for constructing operators from existing ones, typically using composition. Furthermore, an operator may also have a number of additional associated operators, like its (Fréchet) derivative, inverse, and adjoint. Whenever possible, such associated operators are automatically generated when an operator is defined using the operator calculus in ODL, e.g., derivative operators are formed using the chain rule. This is a very powerful part of ODL that reduces the risk for errors and simplifies testing.

Another important part of ODL is its usage of external software libraries for performing specific tasks. When working with tomographic inverse problems, one such task is computing the 2D/3D ray transform and its adjoint. For this ODL employs the ASTRA toolbox [11,13,14], which is a high-performance, GPU accelerated toolbox for tomographic reconstruction freely available from http://www.astra-toolbox.com. The toolbox supports many different data manifolds arising in tomography, including for example circular cone beam, laminography, tomosynthesis, and electron tomography, see [13] for details. It also provides both Matlab and Python interfaces, that expose the core tomographic operations. The latter is used for seamless integration between ODL and ASTRA in the sense that forward and backprojection routines in ASTRA are available as operators in ODL. Likewise, the tomographic data acquisition model in ASTRA is fully reflected by the corresponding data model in ODL. This open up for using ASTRA routines from ODL without unnecessary data copying between GPU and CPU.

The reconstruction methods available in ASTRA are mostly iterative methods. In ODL, on the other hand, one can easily formulate a variational reconstruction algorithm. In this work we will consider ODL/ASTRA to formulate a variational reconstruction algorithm and solve the corresponding optimization problem.

4 Discrete Algebraic Reconstruction

A large class of reconstruction methods for ill-posed inverse problems can be formulated as solving an optimization problem:

$$\min_{f \in X} \Big[\mathcal{L}\big(\mathcal{A}(f), g\big) + \lambda \mathcal{R}(f) \Big]. \tag{3}$$

Here, $\mathcal{R}\colon X \to \mathbb{R}_+$ is the regularization term that accounts for the a priori knowledge by penalizing unfeasible solution candidates, $\mathcal{L}\colon Y \times Y \to \mathbb{R}_+$ is the

data-fit term that quantifies how well two points in the data space agree with each other, and $\lambda > 0$ is the regularization parameter that weights the a priori knowledge against the need to minimize the data-fit term.

Discrete tomography (DT) is a class of tomographic reconstruction methods that are based on the assumption that the unknown object f consists of a few distinct materials, each producing a (almost) constant gray value in the reconstruction. The total variation regularized discrete algebraic reconstruction technique (TVR-DART) is a recent approach that is adapted towards discrete tomography [16,17], which has proven to be more robust than the original DART algorithm [4]. In particular, using the TVR-DART method allows one to significantly improve reconstruction quality and to drastically reduce the number of required projection images and/or exposure to the sample. The idea in TVR-DART is that the image to be recovered is step-function like with a transition between regions that is not necessarily sharp. This is typically the case when one images specimens consisting of only a few different material compositions where each material has a distinct gray value in the corresponding image.

TVR-DART aims to capture such a priori information by combining principles from discrete tomography and compressive sensing. The original formulation in [16,17] uses L^2-norm as the data-fit term, which comes from the assumption that noise in data is additive Gaussian. This is however not always the case, e.g., in HAADF-STEM tomography [8] the noise in data is predominantly Poisson distributed, especially under very low exposure (electron dose) [10].

In the following, we will first formulate TVR-DART in an abstract manner using the language of functional analysis. Next, this abstract version is implemented in ODL/ASTRA using the operator calculus in ODL. Thereby, we can encapsulate all application-specific parts and use high-level ODL solvers that usually expect an operator as argument. In this way, the same problem can be solved with different methods by simply calling different solvers. When a new solver or application-specific code is needed, it needs to be written only once at one place, and can be tested separately. At the lower level of ODL, efficient forward and backward projections are performed using GPU accelerated code provided by the ASTRA toolbox.

In summary, we offer a generic, yet adaptable, version of TVR-DART with a plug-and-play structure that allows one to change the forward operator and the noise model. We demonstrate this flexibility by switching the data-fit term to one that better matches data with Poisson noise, which is more appropriate for tomographic data under low dose conditions.

5 ODL Implementation of TVR-DART

Bearing in mind the functional analytic viewpoint in ODL, our starting point is to formulate the TVR-DART scheme in an abstract setting (Sect. 5.1, Eq. (7)). In the following four sections (Sects. 5.2, 5.3, 5.4 and 5.5) we show how the abstract TVR-DART scheme is implemented using the ODL operator calculus with ASTRA as computational backend for computing projections and corresponding backprojections. In order to emphasis the important points, and due

to space limitations, we have left out some of the code indicated with "`[...]`". The full source code is available at http://github.com/aringh/TVR-DART. We conclude by showing some reconstructions. Note however that the goal of the paper is not to evaluate TVR-DART, for that see [17]. It is to show the flexibility in using ODL/ASTRA as a prototyping tool. Here TVR-DART merely severs as an example of a complex reconstruction method.

5.1 Abstract Formulation

The key assumption in TVR-DART is that the image we seek consists of n gray-scale levels that are separated by a narrow, but smooth, transition layer. Thus, we introduce a (parametrized) segmentation operator $\mathcal{T} : X \times \Theta \to X$ that acts as a kind of segmentation map. It is here given as

$$\mathcal{T}(f,\theta)(x) = \sum_{i=1}^{n-1} (\rho_i - \rho_{i-1}) u_{k_i}\big(f(x) - \tau_i\big) \quad \text{for } x \in \Omega \text{ and } \theta \in \Theta. \tag{4}$$

The parameter space $\Theta := (\mathbb{R} \times \mathbb{R} \times \mathbb{R})^n$ defines the transition characteristics of the n layers (the background ρ_0 is often set to 0). Concretely, $\theta = (\theta_1, \dots, \theta_n) \in \Theta$ with $\theta_i := (\rho_i, \tau_i, K_i)$ where ρ_i is the gray-scale level of the i:th level, τ_i is the mid-point gray-scale value, $k_i := K_i/(\rho_i - \rho_{i-1})$ is the sharpness of the smooth gray-scale transition, and $u\colon \mathbb{R} \to [0,1]$ is the logistic function that models the transition itself

$$u_k(s) := \frac{1}{1 + e^{-2ks}} \quad \text{for } s \in \mathbb{R}. \tag{5}$$

The TVR-DART algorithm for solving (1) is now defined as a method that yields a minimizer to

$$\min_{f \in X,\, \theta \in \Theta} \Big[\mathcal{L}\left([\mathcal{A} \circ \mathcal{T}](f,\theta),\, g\right) + \lambda [\mathcal{S} \circ \mathcal{T}](f,\theta) \Big]. \tag{6}$$

In the above, $\mathcal{L}\colon Y \times Y \to \mathbb{R}_+$ is an appropriate data-fit term and $\mathcal{S}\colon X \to \mathbb{R}_+$ is the regularization. The variant considered in [17] uses a data-fit term $\mathcal{L}(\,\cdot\,, g) = \|\,\cdot - g\|_2^2$ and a regularizing functional $\mathrm{TV}_\varepsilon = [\mathcal{H}_\varepsilon \circ \nabla]$, where $\mathcal{H}_\varepsilon$ is the Huber norm and ∇ is the spatial gradient operator. The Huber norm is a smooth surrogate functional for the L^1-norm, and $\mathcal{H}_\varepsilon \circ \nabla$ is thus a smoothed version of TV. Hence, (6) becomes

$$\min_{f \in X, \theta \in \Theta} \Big[\big\| [\mathcal{A} \circ \mathcal{T}](f,\theta) - g \big\|_2^2 + \lambda [\mathcal{H}_\varepsilon \circ \nabla \circ \mathcal{T}](f,\theta) \Big]. \tag{7}$$

Gradient based methods can be used to solve (7) since its objective functional is smooth. In [17] one such solution method was presented, based on an alternating optimization over f and θ. We take a similar approach here, and to this end define the two operator $\mathcal{T}_\theta : X \to X$, defined by $\mathcal{T}_\theta(f) = \mathcal{T}(f,\theta)$, and $\mathcal{T}_f : \Theta \to X$, defined $\mathcal{T}_f(\theta) = \mathcal{T}(f,\theta)$, where θ and f are seen as fix parameters, respectively. In the current implementation we view the sharpness parameter as fixed, but optimize over gray-scale value and mid-point.

5.2 Defining the Inverse Problem

We begin by defining the reconstruction space $X = L_2(\Omega)$ assuming digitization by uniform sampling in Ω with 320×320 pixels:

```
X = odl.uniform_discr(min_pt=[-200, -200],
    max_pt=[200, 200], shape=[320, 320])
```

Next is to define the forward operator as the ray transform $\mathcal{A} : X \to Y$ in (2).

```
M_angle_part = odl.uniform_partition(0, np.pi, 18, nodes_on_bdry=
    True)
M_detector_part = odl.uniform_partition(-200, 200, 500)
M = odl.tomo.Parallel2dGeometry(M_angle_part, M_detector_part)
A = odl.tomo.RayTransform(X, M, impl='astra_cuda')
```

Note that there is no need to explicitly specify the range Y of the ray transform $\mathcal{A}$, which are functions defined on $\mathbb{M}$ (data manifold). Y is given indirectly by the `geometry`-object, which defines $\mathbb{M}$ through `M_angle_part` for the angular sampling of the lines and `M_detector_part` for the detector sampling. This information is typically provided by the experimental setup.

5.3 Defining the Objective Functional

To define the objective functional in (7), we begin by setting up the soft segmentation operator $\mathcal{T}_\theta : X \to X$ (see Sect. 5.5 for its implementation):

```
T_theta = SoftSegmentationOperator(X, base_value, thresholds,
    values, sharpness)
```

The regularization term $\mathcal{R} : X \to X$ in (7), when optimizing over f, is given by $\mathcal{R}_\theta := \mathcal{H}_\varepsilon \circ \nabla \circ \mathcal{T}_\theta$, which can be implemented using the operator calculus in ODL:

```
gradient = odl.Gradient(X)
gradient = odl.PointwiseNorm(gradient.range) * gradient
H = HuberNorm(X, 0.0001)
R_theta = H * gradient * T_theta
```

In the above, `PointwiseNorm` is used to define the isotropic TV-like term and a description of how to implement the Huber norm is given in Sect. 5.5. Next the data-fit term in (7) as a function $f \mapsto \left\| [\mathcal{A} \circ \mathcal{T}_\theta](f) - g \right\|_2^2$ is implemented as

```
l2_norm = odl.solvers.L2NormSquared(A.range)
l2_norm = l2_norm.translated(data)
data_fit_theta = l2_norm * A * T_theta
```

The `l2_norm.translated(data)` command shifts the origin of the L_2-norm functional, i.e., it changes the functional $\|\cdot\|_2^2$ into $\|\cdot - g\|_2^2$. Hence, the complete objective functional in (7), when optimizing over f, can be assembled as

```
obj_theat = data_fit_theta + reg_param * R_theta
```

The implementation for optimizing over θ is analogous.

5.4 Solving the Optimization Problem

Since the objective functional in (7), both when seen as a functional over f and over θ, is smooth we can use a smooth solver such as limited-memory BFGS [6, Sect. 13.5] with backtracking line-search [6, Sect. 11.5] in the alternating optimization. A BFGS solver and backtracking line-search is built into ODL, and the alternating optimization can thus be implemented as follows.

```
reco = fbp
theta = theta_init
for i in range(10):
  [...]
  linesearch = odl.solvers.BacktrackingLineSearch(obj_theta)
  odl.solvers.bfgs_method(f=obj_theta, x=reco, line_search=
      linesearch, maxiter=10, tol=1e-8, num_store=10)
  [...]
  linesearch = odl.solvers.BacktrackingLineSearch(obj_f)
  odl.solvers.bfgs_method(f=obj_f, x=theta, line_search=
      linesearch, maxiter=2, tol=1e-8, num_store=2)
```

The command `x=reco` specifies the initial iterate for the BFGS solver, and in the first outer iteration is taken as a reconstruction `fbp` obtained from standard filtered backprojection (FBP) using a Hann filter. Example reconstructions using this algorithm are shown later in Fig. 1.

5.5 Implementing the Huber Norm and Soft Segmentation Operator

The Huber norm and soft segmentation operator are not part of ODL and need to be added. They are implemented as an ODL `Functional` and `Operator` object, respectively.

Starting with the Huber norm, its mathematical definition is

$$\mathcal{H}_\varepsilon(f) = \int_\Omega f_\varepsilon(x)dx \quad \text{where} \quad f_\varepsilon(x) = \begin{cases} |f(x)| - \dfrac{\varepsilon}{2} & \text{if } |f(x)| \geq \varepsilon \\ \dfrac{f(x)^2}{2\varepsilon} & \text{if } |f(x)| < \varepsilon. \end{cases}$$

In the ODL implementation below we uses `q_part` to denote the quadratic region, i.e., where $|f(x)| < \varepsilon$.

```
class HuberNorm(Functional):
  [...]
  def _call(self, f):
   """Evaluating the functional."""
    q_part = f.ufuncs.absolute().asarray() < self.epsilon
    q_part = np.float32(q_part)
    f_eps = ((f * q_part)**2 / (2.0 * self.epsilon) +
             (f.ufuncs.absolute() - self.epsilon / 2.0) *
             (1-q_part))
    # This line takes the inner product with the one-function.
    return f_eps.inner(self.domain.one())
```

Since we use a smooth solver, we also need to provide the gradient associated with the Huber norm, which is an element $\nabla \mathcal{H}_\varepsilon(f) \in X$ that satisfies

$$\mathcal{H}'_\varepsilon(f)(h) = \langle \nabla \mathcal{H}_\varepsilon(f), h \rangle_X.$$

In the above, the bounded linear operator $\mathcal{H}'_\varepsilon(f) : X \to \mathbb{R}$ is the Fréchet derivative of $\mathcal{H}_\varepsilon$ at f. For the Huber norm, the gradient at $f \in X$ is

$$\nabla \mathcal{H}_\varepsilon(f)(x) = \frac{\partial}{\partial f} f_\varepsilon(x) = \begin{cases} 1 & \text{if } f(x) \geq \varepsilon \\ -1 & \text{if } f(x) \leq -\varepsilon \\ \dfrac{f(x)}{\varepsilon} & \text{if } |f(x)| < \varepsilon. \end{cases}$$

The above is implemented in ODL as a property of the Huber norm functional:

```
@property
def gradient(self):
  """Gradient operator of the functional."""
  func = self
  class HuberNormGradient(Operator):
  [...]
    def _call(self, f):
      q_part = f.ufuncs.absolute().asarray() < func.epsilon
      q_part = np.float32(q_part)
      f_eps_diff = ((f * q_part) / (func.epsilon) +
                    f.ufuncs.sign() * (1-q_part))
      return f_eps_diff
  return HuberNormGradient()
```

The soft segmentation operator $\mathcal{T}_\theta : X \to X$, implicitly defined in (4), can be implemented in a similar way. Below we compute the Fréchet derivative with respect to f (since this operator is not a functional, it does not have a gradient) that can be implemented in ODL as a property. In ODL the derivative of $\mathcal{T}_\theta$ with respect to the function, at $f \in X$, is itself a linear operator $\mathcal{T}'_\theta(f) : X \to X$ given by

$$\mathcal{T}'_\theta(f)(h)(x) = \sum_{i=1}^{n-1} h(x)(\rho_i - \rho_{i-1}) u'_{k_i}\big(f(x) - \tau_i\big) \quad \text{for } h \in X,$$

where u_k' is the derivative of the logistic function (5)

$$u_k'(t) = \frac{2ke^{-2kt}}{(1+e^{-2kt})^2}.$$

The operator $\mathcal{T}_f : \Theta \to X$ is implemented analogously, where the Fréchet derivative of $\mathcal{T}_f$ with respect to θ can be derived similarly.

5.6 Extension to Handle Data with Poisson Noise

It is well-known that minimizing the Kullback-Leibler (KL) divergence of count data is equivalent to *maximum likelihood* estimation when the noise in data g is Poisson distributed [5]. The original notion of KL divergence comes from information theory, and is thus defined for probability measures. The one used in inverse problems is the generalization below to nonnegative functions:

$$\mathbb{D}_{\mathrm{KL}}(g \,|\, h) = \begin{cases} \displaystyle\int_\Omega \left(g(y) \log\left(\frac{g(y)}{h(y)}\right) + h(y) - g(y) \right) dy & g(y) \geq 0,\ h(y) > 0 \\ +\infty & \text{else.} \end{cases} \tag{8}$$

Noise in low count data is often better modeled using a Poisson distribution rather than an additive Gaussian distribution, and many electron tomography applications are low count data [10]. Hence, using (8) as data-fit term in (6), i.e.,

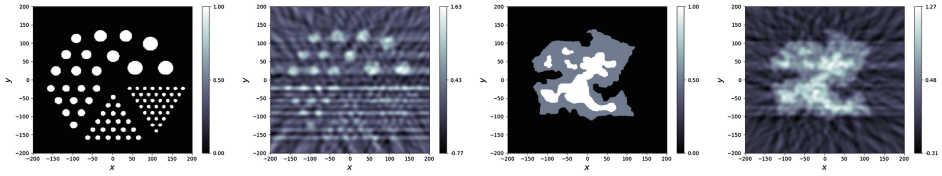
$$\mathcal{L}([\mathcal{A} \circ \mathcal{T}_\theta](f),\, g) = \mathbb{D}_{\mathrm{KL}}\big(g \mid [\mathcal{A} \circ \mathcal{T}_\theta](f)\big),$$

is of interest to applications where the TVR-DART algorithm will be used. Since the KL divergence is already available as a functional in ODL, in order to use KL instead of L_2 we only need to change the data-fit functional:

```
kl = odl.solvers.KullbackLeibler(A.range, prior=data)
data_fit_theta = kl * A * T_theta
```

5.7 Reconstructions

The resulting reconstructions from running the TVR-DART is summarized in Fig. 1, which compares TVR-DART and TV reconstructions, both approaches using L^2-norm and the KL as data-fit term, the former more suitable for data with additive Gaussian noise and the latter more suitable for data with Poisson noise. Tomographic data used for the tests is simulated using both Gaussian and Poisson noise, and for both TV and TVR-DART we have used an FBP reconstruction as an initial starting iterate. In Fig. 1 we also give some figure of merits for the reconstructions, namely Relative Mean Error (RME) (see, e.g., [17, p. 460]) and Structural Similarity index (SSIM) [15].

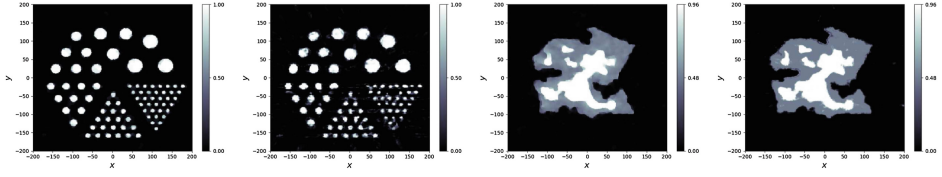


(a) Phantom

(b) Filtered backprojection. RME: 1.96, and SSIM: 0.111.

(c) Phantom

(d) Filtered backprojection. RME: 0.504, and SSIM: 0.303.

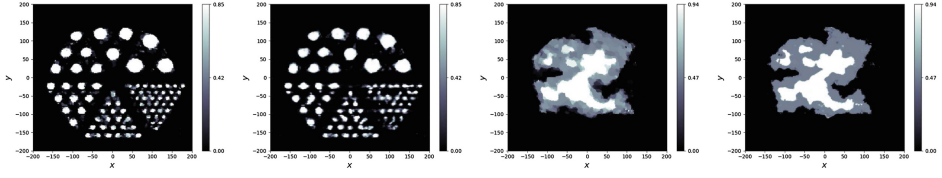


(e) TV with L_2, $\lambda = 3$. RME: 0.184, and SSIM: 0.915.

(f) TVR-DART with L_2, $\lambda = 30$. RME: 0.370, and SSIM: 0.772.

(g) TV with L_2, $\lambda = 10$. RME: 0.109, and SSIM: 0.907.

(h) TVR-DART with L_2, $\lambda = 70$. RME: 0.101, and SSIM: 0.919.



(i) TV with KL, $\lambda = 0.07$. RME: 0.420, and SSIM: 0.829.

(j) TVR-DART with KL, $\lambda = 1$. RME: 0.529, and SSIM: 0.800.

(k) TV with KL, $\lambda = 0.1$. RME: 0.141, and SSIM: 0.894.

(l) TVR-DART with KL, $\lambda = 1$. RME: 0.133, and SSIM: 0.899.

Fig. 1. Phantoms are shown in 1a and 1c. Data is generated from the phantoms and then perturbed by noise. In 1b and 1d, FBP reconstructions from the data with Poisson noise are shown. The reconstructions in the second row, 1e through 1h, are from data with white Gaussian noise, and the reconstructions in the third row, 1i through 1l, are from data with Poisson noise. The images are of size 320×320, and data is acquired from a parallel beam geometry, with 18 equidistant angles between 0 and π and with 500 discretization points on the detector. For data with white Gaussian noise, the noise has a norm that is 5% of the norm of data, and for data with Poisson noise, the data is the outcome of a Poisson distributed variable with parameter given by the noise-free data.

6 Conclusions

We have shown how TVR-DART can be implemented in ODL/ASTRA, and utilizing the modularity and flexibility of ODL we extended the algorithm by changing the data-fit functional $\mathcal{L}$ in (6). In the same way, it is straightforward to change to forward operator $\mathcal{A}$ in (6) in order to use the algorithm in other imaging modalities. As an example, the ODL implementation of TVR-DART can be applied to magnetic resonance imaging by merely changing the forward operator $\mathcal{A}$ to the Fourier transform instead of the ray transform. To conclude, the combination of ODL and ASTRA allows users to specify advanced tomographic reconstruction methods using a high-level mathematical description that facilitates rapid prototyping.

References

1. Adler, J., Kohr, H., Öktem, O.: ODL - a Python framework for rapid prototyping in inverse problems. Royal Institute of Technology (2017) (in Preparation)
2. Alpers, A., Poulsen, H.F., Knudsen, E., Herman, G.T.: A discrete tomography algorithm for improving the quality of three-dimensional X-ray diffraction grain maps. J. Appl. Crystallogr. **39**(4), 582–588 (2006)
3. Batenburg, K.J.: A network flow algorithm for reconstructing binary images from continuous X-rays. J. Math. Imaging Vis. **30**(3), 231–248 (2008)
4. Batenburg, K.J., Sijbers, J.: DART: a practical reconstruction algorithm for discrete tomography. IEEE Trans. Image Process. **20**(9), 2542–2553 (2011)
5. Bertero, M., Lantéri, H., Zanni, L.: Iterative image reconstruction: a point of view. Math. Methods Biomed. Imaging Intensity-Modulated Radiat. Ther. (IMRT) **7**, 37–63 (2008)
6. Griva, I., Nash, S.G., Sofer, A.: Linear and Nonlinear Optimization, 2nd edn. SIAM (2009)
7. Liao, H.Y., Herman, G.T.: A coordinate ascent approach to tomographic reconstruction of label images from a few projections. Disc. Appl. Math. **151**(1), 184–197 (2005)
8. Midgley, P.A., Dunin-Borkowski, R.E.: Electron tomography and holography in materials science. Nat. Mater. **8**(4), 271 (2009)
9. Natterer, F., Wübbeling, F.: Mathematical Methods in Image Reconstruction. SIAM (2001)
10. Öktem, O.: Mathematics of electron tomography. In: Scherzer, O. (ed.) Handbook of Mathematical Methods in Imaging, pp. 937–1031. Springer, New York (2015)
11. Palenstijn, W.J., Batenburg, K.J., Sijbers, J.: Performance improvements for iterative electron tomography reconstruction using graphics processing units (GPUs). J. Struct. Biol. **176**(2), 250–253 (2011)
12. Schüle, T., Schnörr, C., Weber, S., Hornegger, J.: Discrete tomography by convex-concave regularization and DC programming. Disc. Appl. Math. **151**(1), 229–243 (2005)
13. van Aarle, W., Palenstijn, W.J., Cant, J., Janssens, E., Bleichrodt, F., Dabravolski, A., De Beenhouwer, J., Batenburg, K.J., Sijbers, J.: Fast and flexible X-ray tomography using the astra toolbox. Opt. Express **24**(22), 25129–25147 (2016)

14. van Aarle, W., Palenstijn, W.J., De Beenhouwer, J., Altantzis, T., Bals, S., Batenburg, K.J., Sijbers, J.: The ASTRA Toolbox: A platform for advanced algorithm development in electron tomography. Ultramicroscopy **157**, 35–47 (2015)
15. Wang, Z., Bovik, A.C., Sheikh, H.R., Simoncelli, E.P.: Image quality assessment: from error visibility to structural similarity. IEEE Trans. Image Process. **13**(4), 600–612 (2004)
16. Zhuge, X., Jinnai, H., Dunin-Borkowski, R.E., Migunov, V., Bals, S., Cool, P., Bons, A.J., Batenburg, K.J.: Automated discrete electron tomography - towards routine high-fidelity reconstruction of nanomaterials. Ultramicroscopy **175**, 87–96 (2017)
17. Zhuge, X., Palenstijn, W.J., Batenburg, K.J.: TVR-DART: a more robust algorithm for discrete tomography from limited projection data with automated gray value estimation. IEEE Trans. Image Process. **25**(1), 455–468 (2016)