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] Ozan Öktem[1]

[1]Department of Mathematics, KTH Royal Institute of Technology, Stockholm, Sweden.

[2]Computational Imaging, Centrum Wiskunde & Informatica, Amsterdam, The Netherlands.

[3]Mathematical Institute, Leiden University, The Netherlands

September 19, 2017
DGCI2017 - Vienna, Austria

Outline

- Inverse problems
- A new software framework for inverse problems: ODL
- The TVR-DART algorithm for discrete tomography
- Some example code reconstructions

Inverse problems

- Reconstruct a signal $f \in X$ from data $g \in Y$, where

$$g = \mathcal{A}(f_{\mathrm{true}}) \text{ "} + \text{noise"}.$$

- Example: variational reconstruction methods, i.e., formulating the problem as

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

  $\mathcal{L} \colon Y \times Y \to \mathbb{R}_+$ is the data discrepancy term.
  $\mathcal{R} \colon X \to \mathbb{R}_+$ is the regularization term.

- For fixed $\mathcal{A}, \mathcal{L}$, and $\mathcal{R}$ we can use different optimization methods.
  We can apply the same optimization methods for different $\mathcal{A}, \mathcal{L}$, and $\mathcal{R}$.

Inverse problems

- Reconstruct a signal $f \in X$ from data $g \in Y$, where

$$g = \mathcal{A}(f_{\mathrm{true}}) \text{ "} + \text{noise"}.$$

- Example: variational reconstruction methods, i.e., formulating the problem as

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

  $\mathcal{L} \colon Y \times Y \to \mathbb{R}_+$ is the data discrepancy term.
  $\mathcal{R} \colon X \to \mathbb{R}_+$ is the regularization term.
- For fixed $\mathcal{A}, \mathcal{L}$, and $\mathcal{R}$ we can use different optimization methods.
  We can apply the same optimization methods for different $\mathcal{A}, \mathcal{L}$, and $\mathcal{R}$.

Idea: a software framework to facilitate this.

Why a new software framework?

- Multiple modalities: CT, CBCT, PET, SPECT, spectral CT, phase contrast CT, electron tomography, ...
- Mathematical structures/notions: Functional, operator, Fréchet derivative, proximal, diffeomorphism, discretization, sparsifying transforms, ...
- Flexibility: Mathematical structures/notions re-usable across modalities
  $\leadsto$ Make it easy to "play around" with new ideas and combine concepts.
- Collaborative research: Need to share implementations of common concepts
- Reproducible research: Not enough to share theory and pseudocode, also need to share data and concrete implementations
  $\leadsto$ Software components need to be usable by others.
  (Code for this paper is on `github`).

Why a new software framework?

Requirements on a software framework:

- Allow formulation and solution of inverse problems in a common language.
- Make implementations re-usable and extendable.
- Enable fast prototyping on clinically relevant data.
- Leverage the power of existing libraries.
  $\rightsquigarrow$ For example, use ASTRA [Aar16] for computing the Ray transform.

[Aar16] W. van Aarle, et al. Fast and flexible X-ray tomography using the ASTRA toolbox. Optics express, 24(22), 25129-25147 (2016).

Why a new software framework?

Requirements on a software framework:

- Allow formulation and solution of inverse problems in a common language.
- Make implementations re-usable and extendable.
- Enable fast prototyping on clinically relevant data.
- Leverage the power of existing libraries.
  $\rightsquigarrow$ For example, use ASTRA [Aar16] for computing the Ray transform.

Initial situation: No existing framework fit our purpose.

[Aar16] W. van Aarle, et al. Fast and flexible X-ray tomography using the ASTRA toolbox. Optics express, 24(22), 25129-25147 (2016).

Design principles:

- Modularity:
  $\rightsquigarrow$ (almost) freely exchangeable "modules" in the mathematical formulation
  $\rightsquigarrow$ Mathematics as strong guideline for software design
- Abstraction and compartmentalization:
  $\rightsquigarrow$ Separates the "what" (abstract interface) of an object class from the "how" (concrete implementation)
  $\rightsquigarrow$ Makes functions and classes individually testable

Discrete tomography and TVR-DART

An example to illustrate the flexibility.

- Discrete tomography: base on assumption that f_{true} consists of few distinct materials, each producing a constant gray value.
- TVR-DART: variational regularization scheme for discrete tomography [Zhu16]. Key components:
  - The spacial gradient operator $\nabla : X \to X^d$.
  - The Huber norm $\mathcal{H}_\varepsilon : X \to \mathbb{R}_+$,

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

[Zhu16] X. Zhuge, W.J. Palenstijn, K.J. Batenburg. TVR-DART: a more robust algorithm for discrete tomography from limited projection data with automated gray value estimation. IEEE Transactions on Image Processing, 25(1), 455-468 (2016).

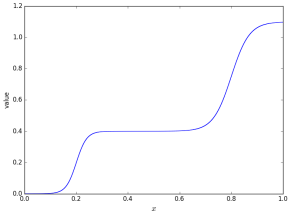
Discrete tomography and TVR-DART

An example to illustrate the flexibility.

- Discrete tomography: base on assumption that f_{true} consists of few distinct materials, each producing a constant gray value.
- TVR-DART: variational regularization scheme for discrete tomography [Zhu16]. Key components:
  - a (parametrized) soft segmentation operator $\mathcal{T} : X \times \Theta \to X$ where $\Theta = (\mathbb{R} \times \mathbb{R} \times \mathbb{R})^n$, $\theta = (\theta_1, \dots, \theta_n)$, and $\theta_i = (\rho_i, \tau_i, k_i)$.

$$\mathcal{T}(f, \theta)(x) = \sum_{i=1}^{n-1} (\rho_i - \rho_{i-1}) u_{k_i}\big(f(x) - \tau_i\big)$$

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



[Zhu16] X. Zhuge, W.J. Palenstijn, K.J. Batenburg. TVR-DART: a more robust algorithm for discrete tomography from limited projection data with automated gray value estimation. IEEE Transactions on Image Processing, 25(1), 455-468 (2016).

The TVR-DART algorithm

- Idea is to estimate both parameters θ and reconstruction f.

$$\min_{f \in X,\, \theta \in \Theta} \Big[\mathcal{L}\left([\mathcal{A} \circ \mathcal{T}](f, \theta),\, g \right) + \lambda [\mathcal{H}_\varepsilon \circ \nabla \circ \mathcal{T}](f, \theta) \Big].$$

- In the original paper [Zhu16], the L_2-norm was used as data discrepancy term: $\mathcal{L}(\cdot, g) = \| \cdot - g \|_2^2$. Optimization problem solved by alternatingly optimize over f and θ using a gradient-based solver. Notation: $\mathcal{T}_\theta : X \to X$ and $\mathcal{T}_f : \Theta \to X$.
- We demonstrate the flexibilty of ODL by showing that this can easily be changed to the Kullback-Leibler functional, which is more suitable if noise is Poisson.

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

[Zhu16] X. Zhuge, W.J. Palenstijn, K.J. Batenburg. TVR-DART: a more robust algorithm for discrete tomography from limited projection data with automated gray value estimation. IEEE Transactions on Image Processing, 25(1), 455-468 (2016).

The TVR-DART algorithm

Components not already in ODL: i) the Huber norm, ii) the soft segmentation operator. Example code implementing the Huber norm:

```
class HuberNorm(Functional):
  [...]
  def _call(self, f):
    """Evaluating the functional."""
    q_part = f.ufuncs.absolute().asarray() < self.epsilon
    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())
  [...]
```

The TVR-DART algorithm

Setting up and solving the optimization problem:

- Defining the tomography problem

```
X = odl.uniform_discr(min_pt=[-200, -200], max_pt=[200, 200],
                      shape=[320, 320])
Y_angle_part = odl.uniform_partition(0, np.pi, 18)
Y_detector_part = odl.uniform_partition(-200, 200, 500)
Y = odl.tomo.Parallel2dGeometry(Y_angle_part, Y_detector_part)
A = odl.tomo.RayTransform(X, Y, impl='astra_cuda')
```

The TVR-DART algorithm

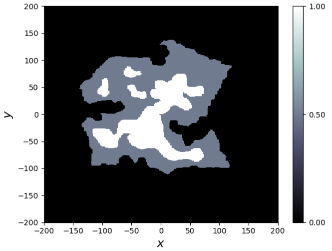
Setting up and solving the optimization problem:

- Defining the cost function

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

# Defining the regularization term
gradient = odl.Gradient(X)
point_norm = odl.PointwiseNorm(gradient.range)
H = HuberNorm(X, 0.0001)
R_theta = H * point_norm * gradient * T_theta

# Defining the data discrepancy term
l2_norm = odl.solvers.L2NormSquared(A.range)
l2_norm = l2_norm.translated(data)
data_fit_theta = l2_norm * A * T_theta

obj_theat = data_fit_theta + reg_param * R_theta
```

- Alternating optimization done with BFGS-method.

Simulation results
Additive white Gaussian noise and L_2 data discrepancy



(a) Phantom (b) TVR-DART with L_2. (c) TV with L_2.

Figure: Reconstructions from data with white Gaussian noise.

Using KL instead of L_2 as data discrepancy term

Difference between using L_2-norm and KL as data discrepancy functional?

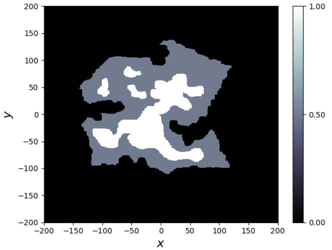
In code we change

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

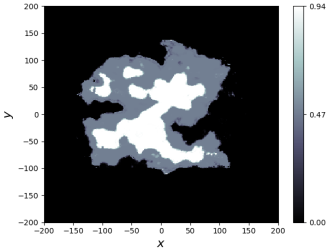
to

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

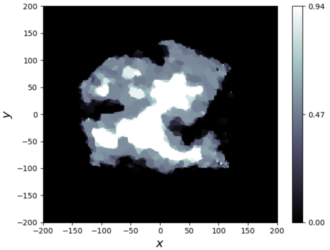
Simulation results
Poisson noise and KL data discrepancy



(a) Phantom



(b) TVR-DART with KL.



(c) TV with KL.

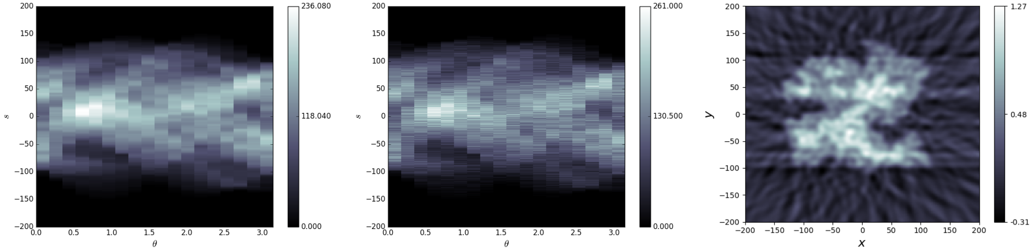
Figure: Reconstructions from data with Poisson noise.

Conclusions

- Introduced a new software framework for fast high-level prototyping of solution methods for inverse problems - ODL (https://github.com/odlgroup/odl).
- Demonstrated the capabilities for fast high-level prototyping, by implementing and extending the TVR-DART algorithm for discrete tomography.

Thank you!

Questions?



(a) Singoram (b) Noisy sinogram. (c) FBP reconstruction.

Figure: Data (sinograms) and FBP reconstruction from data with Poisson noise.