PySAP-ComSET: an accelerated Python package for compressed sensing electron tomography (CS-ET) reconstruction

JyhMiin Lin [1] Martin Jacob [1] Zineb Saghi [1]
Philippe Ciuciu [2] Jean-Luc Starck [3]

[1]Leti, CEA Grenoble

[2]Neurospin, CEA Paris-Saclay

[3]CosmoStat, CEA Paris-Saclay

IWOCL '20: International Workshop on OpenCL Munich Germany, April, 2020

Overview

1. Introduction
   - Compressed sensing for electron tomography
   - Python Sparse data Analysis Package (PySAP)
   - Graphic processing Units (GPU)
2. Development
   - Sparse transform (I): Variational Type: TV, TGV
   - Sparse transform (II): Wavelet
   - Sparse transform (III): Mixed type: Ridgelet, wt_tv, polar_tv
   - Optimisation algorithms
3. To-do list
   - Given an image, empirically determine the best parameter and sparse transform for the compressed sensing problem
   - Replace the generic primal-dual algorithms by others in ModOpt
   - Test X-ray tomography and spectro-tomography
   - Move the optimisation algorithm to GPU?

Introduction

- Compressed sensing electron tomography using Python
- Python Sparse data Analysis Package (PySAP)
- Graphic processing Units (GPU)

Compressed sensing electron tomography using Python

- Purpose: to accelerate the image acquisition with a reduced number of projections
- Solving the minimisation problem:

$$x = \arg\min_{x} \|y - \mathbf{A}x\|_2 + \lambda|\Psi x| \tag{1}$$

x: image
y: sinogram
$\mathbf{A}$: Radon transform
Ψ: sparse transform

- Previous compressed sensing electron tomography: TV [1], Total nuclear norm [2], HOTV [3], TGV [4], wavelets (Jacob -)
- Python: Numpy/Scipy is the current standard tool in data science community.

Python Sparse data Analysis Package (PySAP)

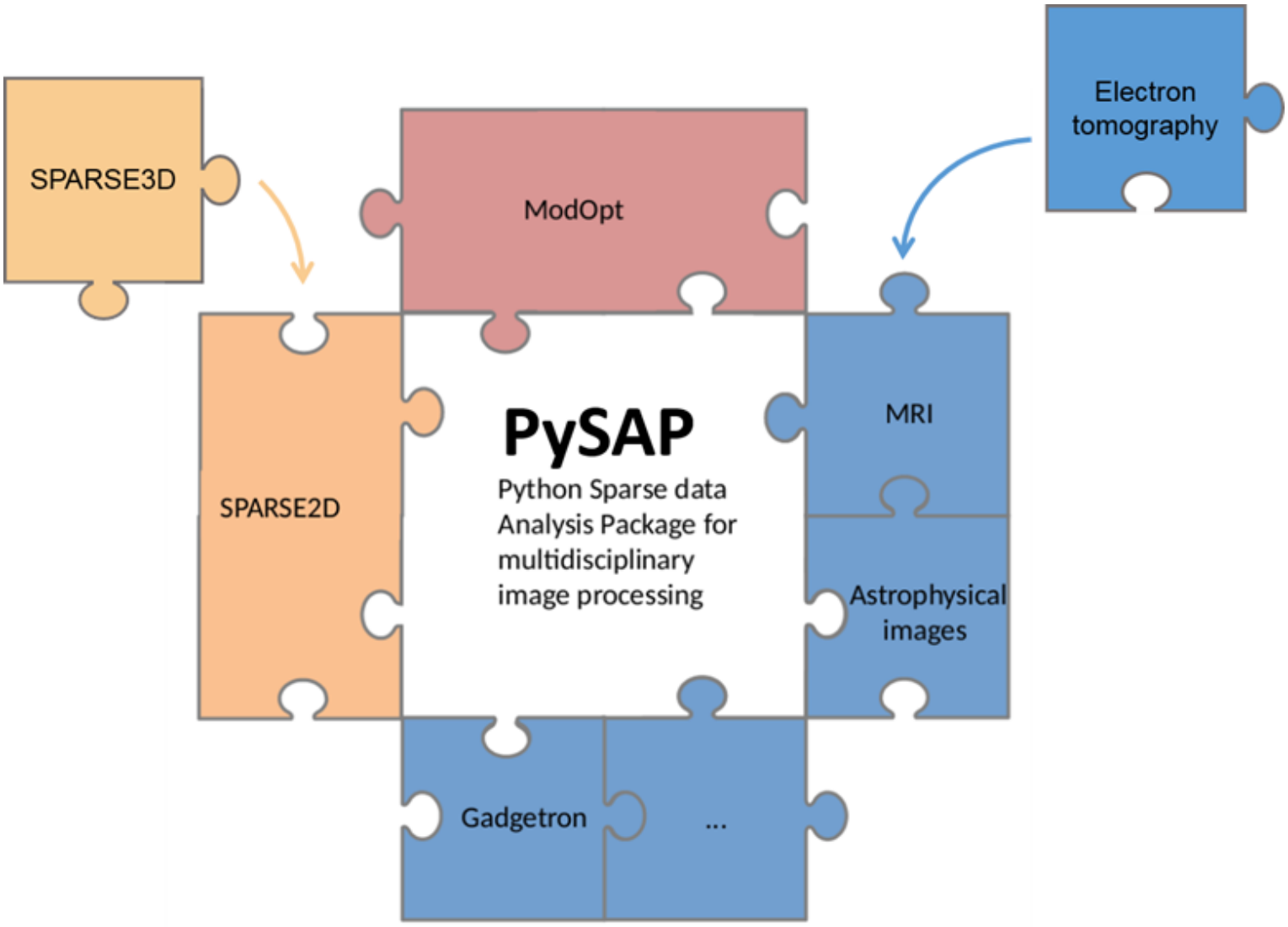

Figure: ET as an application of PySAP

Graphics processing units (GPUs)

- Massive parallelism, but **hard** to be integrated into Python
- GPU-powered Radon transform Python packages are available
  - astra-toolbox [5]: using CUDA and ctype; the installation is tricky
  - Huber 2019 [4]: a script which accelerates TGV using PyOpenCL
  - Fourier domain using PyNUFFT [6]: PyCUDA and PyOpenCL ready. Now I have created the PyNUFFT-based 2D Radon transform (including numpy, PyCUDA and PyOpenCL).

UNMATCHED POWER. UNMATCHED CREATIVE FREEDOM. NVIDIA® QUADRO® P4000

Extreme Visual Computing Performance in a Single Slot Form Factor.

The NVIDIA Quadro P4000 combines a 1792 CUDA core Pascal GPU, large 8 GB GDDR5 memory and advanced display technologies to deliver the performance and features that are required by demanding professional applications. The ability to create an expansive visual workspace of up to four 5K displays (5120x2880 @ 60Hz) with HDR color support lets you view your creations in stunning detail. The P4000 is specially designed with the performance that is necessary to drive immersive VR environments. Additionally, you can create massive digital signage solutions of up to thirty-two 4K displays per system by connecting multiple P4000s via Quadro Sync II[2].

Quadro cards are certified with a broad range of

FEATURES

- Four DisplayPort 1.4 Connectors[1]
- DisplayPort with Audio
- 3D Stereo Support with Stereo Connector[1]
- NVIDIA GPUDirect™ Support
- Quadro Sync II[2] Compatibility
- NVIDIA nView® Desktop Management Software
- HDCP 2.2 Support
- NVIDIA Mosaic[3]
- Dedicated hardware video encode and decode engines[4]

SPECIFICATIONS

GPU Memory	**8 GB GDDR5**
Memory Interface	**256-bit**
Memory Bandwidth	**Up to 243 GB/s**
NVIDIA CUDA® Cores	**1792**
System Interface	**PCI Express 3.0 x16**
Max Power Consumption	**105 W**
Thermal Solution	**Active**
Form Factor	**4.4" H x 9.5" L, Single Slot, Full Height**

Development

- Sparse transform (I): Variational Type: TV, TGV
- Sparse transform (II): Wavelet
- Sparse transform (III): Mixed type: Ridgelet, wt_tv, polar_tv
- Optimisation algorithms

Sparse transform (I): Variational Type: TV, TGV

- Any variational type sparse transform can be represented as a scipy.sparse.linalg.LinearOperator.
  - TV: isotropic and anisotropic
  - TGV: total generalised variation
- Easily plugged into the TV-like optimisation algorithms.
- However, isotropic TV uses the root-sum-squared of the directional image gradients, which is non-linear.
- $s = \sqrt{sx^2 + sy^2}$ can be easily coded before the non-linear saturation function
- Refer to the comset.linalg.radon2d.tv_iso function
- I use Equation (34) in [7]

Sparse transform (II): Wavelet

- Decimated: using the wavedecn and waverecn in Pywavelets, but shift-variant
- Undecimated: the stationary (à trous) algorithm: using the swtn and iswtn
- It just works using parameters: 'bior4.4', level = 4, norm=False, trim_approx=True
- For now, the simple primal-dual algorithm (Equation (15) in [7]) is used.
- The optimisation may converge faster with ModOpt.

Sparse transform (III): Mixed type: Ridgelet, wt_tv, polar_tv

- No definite conclusion but the following mixed types have been implemented:
  - Ridgelet transform: originally proposed for **denoising**. 1D wavelet along the radial direction in the Radon domain (See comset.linalg.radon.ridgelet)
  - wt_tv: Plug the simple primal-dual shrinkage thresholding into the tv regularization (See comset.linalg.radon.wt_tv)
  - polar_tv: apply the anisotropic TVs in the real image domain and the Radon domain. (See comset.linalg.radon.polar_tv)

Optimisation algorithms

- Loris and Verhoeven [7] explicitly wrote two algorithms for variational type and wavelets type regularizations.
- "variational type": No inverse transform exists for the TV or TGV
- "wavelet type": The inverse transform exists for a wavelet transform

I have tried Equation (15) for wavelet type and Equation (34) for variational type

To-do list

- Given an image, empirically determine the best parameter and sparse transform for the compressed sensing problem
- Replace the generic primal-dual algorithms by the ones in ModOpt
- Test X-ray tomography and spectro-tomography
- Move the optimisation algorithm to GPU?

References I

Zineb Saghi, Daniel J Holland, Rowan Leary, Andrea Falqui, Giovanni Bertoni, Andrew J Sederman, Lynn F Gladden, and Paul A Midgley.
Three-dimensional morphology of iron oxide nanoparticles with reactive concave surfaces. a compressed sensing-electron tomography (CS-ET) approach.
Nano letters, 11(11):4666–4673, 2011.

Zhichao Zhong, Willem Jan Palenstijn, Jonas Adler, and K Joost Batenburg.
Eds tomographic reconstruction regularized by total nuclear variation joined with HAADF-STEM tomography.
Ultramicroscopy, 191:34–43, 2018.

References II

Toby Sanders, Anne Gelb, Rodrigo B Platte, Ilke Arslan, and Kai Landskron.
Recovering fine details from under-resolved electron tomography data using higher order total variation l1 regularization.
Ultramicroscopy, 174:97–105, 2017.

Richard Huber, Georg Haberfehlner, Martin Holler, Gerald Kothleitner, and Kristian Bredies.
Total generalized variation regularization for multi-modal electron tomography.
Nanoscale, 11(12):5617–5632, 2019.

References III

Wim van Aarle, Willem Jan Palenstijn, Jan De Beenhouwer, Thomas Altantzis, Sara Bals, K Joost Batenburg, and Jan Sijbers.
The ASTRA toolbox: A platform for advanced algorithm development in electron tomography.
Ultramicroscopy, 157:35–47, 2015.

Jyh-Miin Lin.
Python non-uniform fast fourier transform (PyNUFFT): An accelerated non-Cartesian MRI package on a heterogeneous platform (CPU/GPU).
Journal of Imaging, 4(3):51, 2018.

References IV


Ignace Loris and Caroline Verhoeven.
Iterative algorithms for total variation-like reconstructions in seismic tomography.
GEM-International Journal on Geomathematics, 3(2):179–208, 2012.