12.Cartesian_Calibrationless_CS-pMRI_recon

January 7, 2021

1 Twelveth exercice: Calibrationless CS-pMR image reconstruction from undersampled Cartesian data

In this tutorial we will reconstruct a 2D MR image from multicoil Cartesian under-sampled kspace measurements.

We use the toy datasets available in pysap, more specifically a 2D brain slice and under-sampled Cartesian acquisition over 32 channels. We compare zero-order image reconstruction with **calibrationless** multi-coil Compressed sensing reconstructions (analysis vs synthesis formulation) using the FISTA algorithm for the synthesis formulation and the Condat-Vu algorithm for the analysis formulation. **Structured sparsity** will be promoted in the wavelet domain, using either Symmlet-8 (analysis and synthesis) or undecimated bi-orthogonal wavelets (analysis only) considering group-LASSO or OSCAR-based regularization. The multicoil data $(y_\ell)_\ell$ is collected across multiple, say L, channels.

We remind that the synthesis formulation of the Calobrationless CS-PMRI problem reads (minimization in the sparsifying domain):

$$\widehat{Z} = \arg \min_{Z \in C^{n_\Psi \times L}} \frac{1}{2} \sum_{\ell=1}^{L} \|y_\ell - \Omega F \Psi^* z_\ell\|_2^2 + \lambda \mathcal{R}(Z)$$

where $Z = [z_1, \ldots, z_L]$ and $X = [x_1, \ldots, x_L] \in C^{n \times L}$ such that $x_l = \Psi^* z_l$. The image solution is given by $\widehat{x} = \Psi^* \widehat{z}$. For an orthonormal wavelet transform, we have $n_\Psi = n$ while for a frame we may have $n_\Psi > n$. The regularization term promotes structured sparsity. For instance when one chooses group-LASSO regularization $\mathcal{R}(Z) = \sum_{i=1}^{n_\Psi} \|z_i\|_2$, where the L2 norm involves the L channels per wavelet coefficient z_i.

The analysis formulation consists in minimizing the following cost function (min. in the image domain):

$$\widehat{X} = \arg \min_{X \in C^{n \times L}} \frac{1}{2} \sum_{\ell=1}^{L} \|y_\ell - \Omega F x_\ell\|_2^2 + \lambda \mathcal{R}(\Psi X)\,.$$

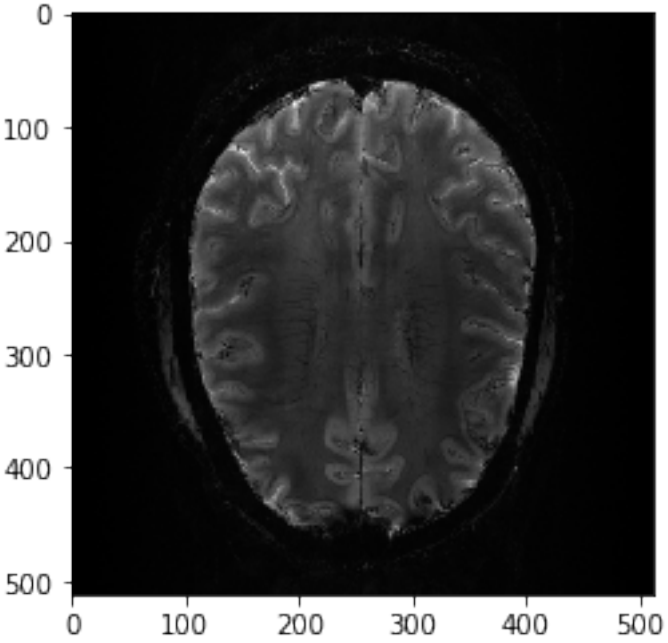
- Author: Chaithya G R & Philippe Ciuciu
- Date: 01/07/2021
- Target: ATSI MSc students, Paris-Saclay University

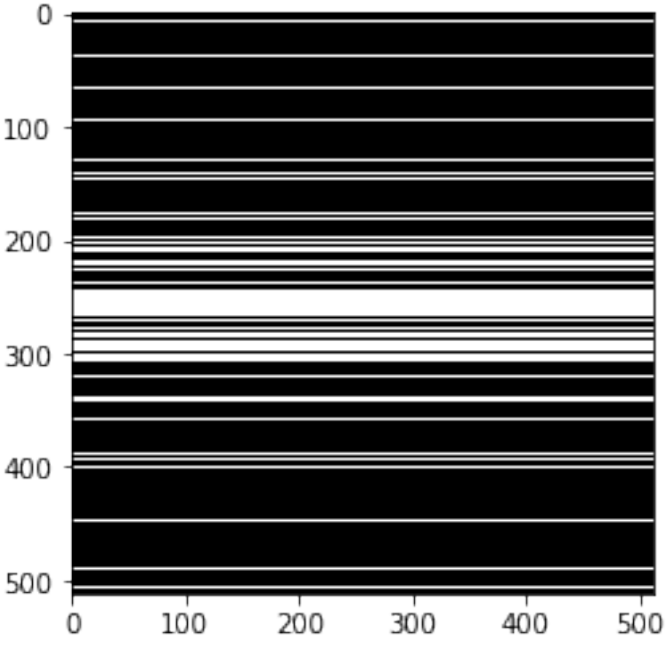
[1]:
```
from mri.operators import FFT, WaveletN, WaveletUD2, OWL
from mri.operators.utils import convert_mask_to_locations
from mri.reconstructors import CalibrationlessReconstructor
```

```
import pysap
from pysap.data import get_sample_data

# Third party import
from modopt.math.metrics import ssim
from modopt.opt.linear import Identity
from modopt.opt.proximity import GroupLASSO
import numpy as np
import matplotlib.pyplot as plt
```

```
/home/ciuciu/anaconda3/lib/python3.7/site-
packages/mri/operators/fourier/cartesian.py:33: UserWarning: pynufft python
package has not been found. If needed use the master release. Till then you
cannot use NUFFT on GPU
  warnings.warn("pynufft python package has not been found. If needed use "
/home/ciuciu/anaconda3/lib/python3.7/site-
packages/mri/operators/fourier/non_cartesian.py:42: UserWarning: gpuNUFFT python
package has not been found. If needed please check on how to install in README
  warnings.warn("gpuNUFFT python package has not been found. If needed "
```

[2]:
```
%matplotlib inline

cartesian_ref_image = get_sample_data('2d-pmri')
#cartesian_ref_image = np.sum(np.reshape(cartesian_ref_image,(8,4,512,512)),␣
 ↪axis=0)
image = pysap.Image(data=np.sqrt(np.sum(cartesian_ref_image.data**2, axis=0)))
#image = pysap.Image(data=np.sqrt(np.sum(cartesian_ref_image**2, axis=0)))
# Obtain MRI non-cartesian mask
mask = get_sample_data("cartesian-mri-mask")
kspace_loc = convert_mask_to_locations(mask.data)
plt.figure()
plt.imshow(image, cmap='gray')
plt.figure()
plt.imshow(mask, cmap='gray')
plt.show()
```


0
100
200
300
400
500
0
100
200
300
400
500


0
100
200
300
400
500
0
100
200
300
400
500

1.1 Generate the kspace

From the 2D brain slice and the acquisition mask, we retrospectively undersample the k-space using a cartesian acquisition mask We then reconstruct the zero order solution as a baseline

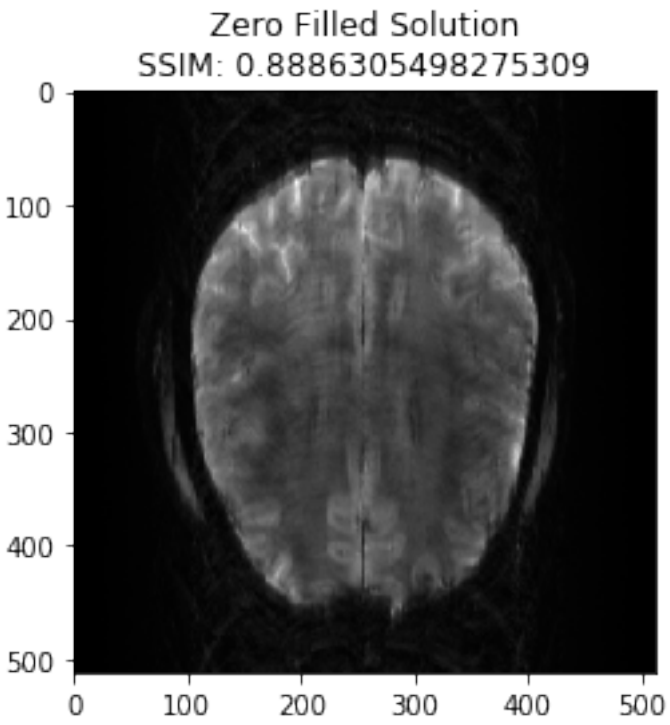
Get the locations of the kspace samples

[3]:
```
# Get the locations of the kspace samples and the associated observations
fourier_op = FFT(samples=kspace_loc, shape=image.shape,
                 n_coils=cartesian_ref_image.shape[0])
kspace_obs = fourier_op.op(cartesian_ref_image)
```

```
/home/ciuciu/anaconda3/lib/python3.7/site-
packages/mri/operators/fourier/utils.py:76: FutureWarning: Using a non-tuple
sequence for multidimensional indexing is deprecated; use `arr[tuple(seq)]`
instead of `arr[seq]`. In the future this will be interpreted as an array index,
`arr[np.array(seq)]`, which will result either in an error or a different
result.
  mask[test] = 1
```

Zero order solution

[4]:
```
zero_filled = fourier_op.adj_op(kspace_obs)
image_rec0 = pysap.Image(data=np.sqrt(np.sum(np.abs(zero_filled)**2, axis=0)))
# image_rec0.show()
base_ssim = ssim(image_rec0, image)
plt.imshow(np.abs(image_rec0), cmap='gray')
# Calculate SSIM
base_ssim = ssim(image_rec0, image)
plt.title('Zero Filled Solution\nSSIM: {}'.format(base_ssim))
plt.show()
```



1.2 Synthesis formulation: FISTA vs POGM optimization

We now want to refine the zero order solution using a FISTA optimization. The cost function is set to Proximity Cost + Gradient Cost

[5]:
```
linear_op = WaveletN(
    wavelet_name='sym8',
    nb_scale=4,
    n_coils=cartesian_ref_image.shape[0],
)
#padding_mode="periodization"

regularizer_op = GroupLASSO(6e-8)
```

1.3 Setup reconstructor:

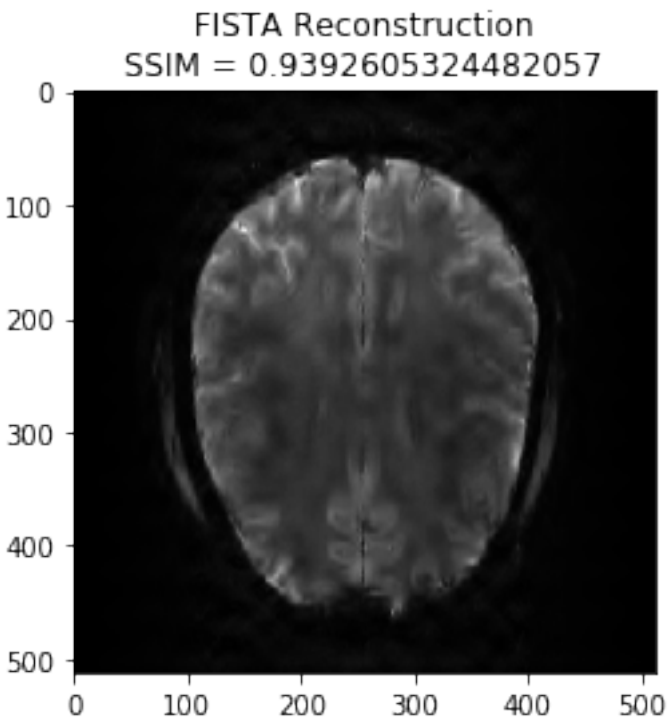
[25]:
```
reconstructor = CalibrationlessReconstructor(
    fourier_op=fourier_op,
    linear_op=linear_op,
    regularizer_op=regularizer_op,
    gradient_formulation='synthesis',
```

```
    verbose=1,
)
```

```
WARNING: Making input data immutable.

Lipschitz constant is 1.1000000000000556
The lipschitz constraint is satisfied
```

[27]:
```
x_final, costs, metrics = reconstructor.reconstruct(
    kspace_data=kspace_obs,
    optimization_alg='fista',
    num_iterations=100,
)
image_rec = pysap.Image(data=np.sqrt(np.sum(np.abs(x_final)**2, axis=0)))
recon_ssim = ssim(image_rec, image)

plt.imshow(np.abs(image_rec), cmap='gray')
plt.title('FISTA Reconstruction\nSSIM = ' + str(recon_ssim))
plt.show()
```

```
 - mu:  6e-08
 - lipschitz constant:  1.1000000000000556
 - data:  (512, 512)
 - wavelet:  <mri.operators.linear.wavelet.WaveletN object at 0x7f0fa9102f50> -
4
 - max iterations:  100
 - image variable shape:  (512, 512)
 - alpha variable shape:  (32, 291721)
----------------------------------------
Starting optimization…

100% (100 of 100) |######################| Elapsed Time: 0:05:22 Time:  0:05:22

 - final iteration number:  100
 - final log10 cost value:  6.0
 - converged:  False
Done.
Execution time:  326.28968491905835  seconds
----------------------------------------
```

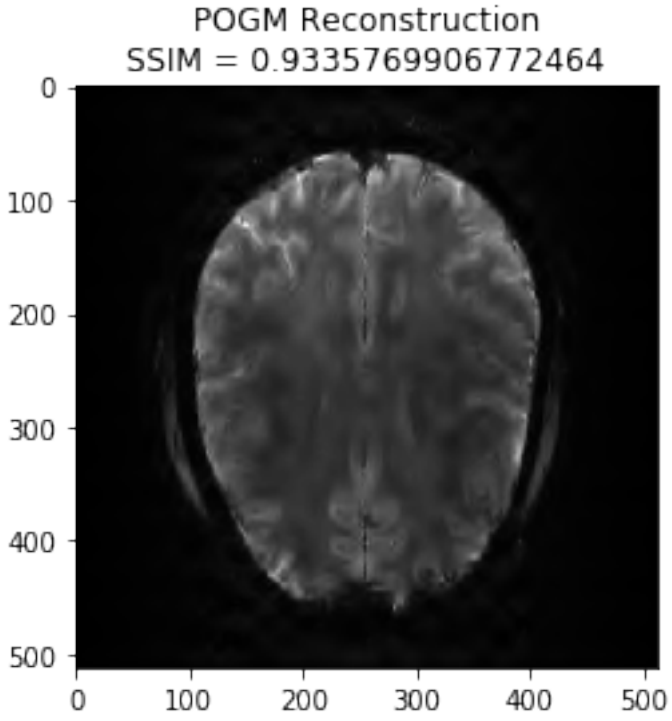


1.4 POGM optimization

[28]:
```
x_final, costs, metrics = reconstructor.reconstruct(
    kspace_data=kspace_obs,
    optimization_alg='pogm',
    num_iterations=100,
)
image_rec = pysap.Image(data=np.sqrt(np.sum(np.abs(x_final)**2, axis=0)))
recon_ssim = ssim(image_rec, image)

plt.imshow(np.abs(image_rec), cmap='gray')
plt.title('POGM Reconstruction\nSSIM = ' + str(recon_ssim))
plt.show()
```

```
 - mu:  6e-08
 - lipschitz constant:  1.1000000000000556
 - data:  (512, 512)
 - wavelet:  <mri.operators.linear.wavelet.WaveletN object at 0x7f0fa9102f50> -
4
 - max iterations:  100
 - image variable shape:  (32, 512, 512)
----------------------------------------
```

```
Starting optimization…
100% (100 of 100) |######################| Elapsed Time: 0:07:16 Time:  0:07:16
 - final iteration number:  100
 - final log10 cost value:  6.0
 - converged:  False
Done.
Execution time:  443.8226826849859  seconds
----------------------------------------
```



1.5 Analysis formulation: Condat-Vu reconstruction

[32]:
```
linear_op = WaveletN(
    wavelet_name='sym8',
    nb_scale=4,
    n_coils=cartesian_ref_image.shape[0],
)
#padding_mode="periodization"

regularizer_op = GroupLASSO(6e-8)
```

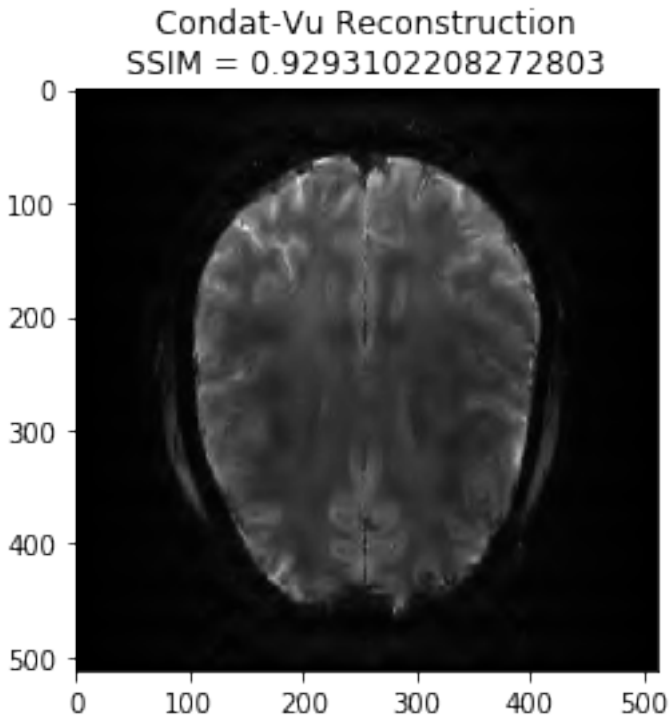
[33]:
```
reconstructor = CalibrationlessReconstructor(
    fourier_op=fourier_op,
    linear_op=linear_op,
    regularizer_op=regularizer_op,
    gradient_formulation='analysis',
    verbose=1,
)
```

```
WARNING: Making input data immutable.

Lipschitz constant is 1.1
The lipschitz constraint is satisfied
```

[34]:
```
x_final, costs, metrics = reconstructor.reconstruct(
    kspace_data=kspace_obs,
    optimization_alg='condatvu',
    num_iterations=100,
)

image_rec = pysap.Image(data=np.sqrt(np.sum(np.abs(x_final)**2, axis=0)))
recon_ssim = ssim(image_rec, image)

plt.imshow(np.abs(image_rec), cmap='gray')
plt.title('Condat-Vu Reconstruction\nSSIM = ' + str(recon_ssim))
plt.show()
```

```
 - mu:  6e-08
 - lipschitz constant:  1.1
 - tau:  0.9523809433107514
 - sigma:  0.5
 - rho:  1.0
 - std:  None
 - 1/tau - sigma||L||^2 >= beta/2:  True
 - data:  (512, 512)
 - wavelet:  <mri.operators.linear.wavelet.WaveletN object at 0x7f0fa8f50f10> -
4
 - max iterations:  100
 - number of reweights:  0
 - primal variable shape:  (32, 512, 512)
 - dual variable shape:  (32, 291721)
----------------------------------------
Starting optimization...

100% (100 of 100) |######################| Elapsed Time: 0:10:51 Time:  0:10:51

 - final iteration number:  100
 - final cost value:  1000000.0
 - converged:  False
Done.
```

```
Execution time:  657.6715358300135  seconds
----------------------------------------
```



[35]:
```
# Undecimated Wavelets
#linear_op = WaveletUD2(
#    wavelet_id=24,
#    nb_scale=4,
#    n_coils=cartesian_ref_image.shape[0],
#)
```

[6]:
```
#regularizer_op = GroupLASSO(6e-8)
coeffs = linear_op.op(cartesian_ref_image)
regularizer_op = OWL(
    alpha=1.05e-8,
    beta=0,
    mode='band_based',
    n_coils=cartesian_ref_image.shape[0],
    bands_shape=linear_op.coeffs_shape,
)
```

[7]:
```
reconstructor = CalibrationlessReconstructor(
    fourier_op=fourier_op,
    linear_op=linear_op,
    regularizer_op=regularizer_op,
```

```
    gradient_formulation='analysis',
    verbose=1,
)
```

```
WARNING: Making input data immutable.

Lipschitz constant is 1.1
The lipschitz constraint is satisfied
```

[8]:
```
x_final, costs, metrics = reconstructor.reconstruct(
    kspace_data=kspace_obs,
    optimization_alg='condatvu',
    num_iterations=100,
)

image_rec = pysap.Image(data=np.sqrt(np.sum(np.abs(x_final)**2, axis=0)))
recon_ssim = ssim(image_rec, image)

plt.imshow(np.abs(image_rec), cmap='gray')
plt.title('Condat-Vu Reconstruction\nSSIM = ' + str(recon_ssim))
plt.show()
```

```
WARNING: Making input data immutable.

 - mu:  [<modopt.opt.proximity.OrderedWeightedL1Norm object at 0x7fb491f31c50>,
<modopt.opt.proximity.OrderedWeightedL1Norm object at 0x7fb491f8fc90>,
<modopt.opt.proximity.OrderedWeightedL1Norm object at 0x7fb45fc62090>,
<modopt.opt.proximity.OrderedWeightedL1Norm object at 0x7fb45fc62950>,
<modopt.opt.proximity.OrderedWeightedL1Norm object at 0x7fb45fc62f90>,
<modopt.opt.proximity.OrderedWeightedL1Norm object at 0x7fb45fc62d90>,
<modopt.opt.proximity.OrderedWeightedL1Norm object at 0x7fb45fc62250>,
<modopt.opt.proximity.OrderedWeightedL1Norm object at 0x7fb45fc62a90>,
<modopt.opt.proximity.OrderedWeightedL1Norm object at 0x7fb45fc62110>,
<modopt.opt.proximity.OrderedWeightedL1Norm object at 0x7fb45fc62910>,
<modopt.opt.proximity.OrderedWeightedL1Norm object at 0x7fb45fc62750>,
<modopt.opt.proximity.OrderedWeightedL1Norm object at 0x7fb491ffa990>,
<modopt.opt.proximity.OrderedWeightedL1Norm object at 0x7fb45ff830d0>]
 - lipschitz constant:  1.1
 - tau:  0.9523809433107514
 - sigma:  0.5
 - rho:  1.0
 - std:  None
 - 1/tau - sigma||L||^2 >= beta/2:  True
 - data:  (512, 512)
 - wavelet:  <mri.operators.linear.wavelet.WaveletN object at 0x7fb45ff9d2d0> -
4
 - max iterations:  100
 - number of reweights:  0
 - primal variable shape:  (32, 512, 512)
```

```
 - dual variable shape:  (32, 291721)
----------------------------------------
Starting optimization…

100% (100 of 100) |######################| Elapsed Time: 0:07:40 Time:  0:07:40

 - final iteration number:  100
 - final cost value:  1000000.0
 - converged:  False
Done.
Execution time:  466.3144229580648  seconds
----------------------------------------
```