08.Cartesian_CS_image_recon

January 6, 2021

1 Eighth exercice: MR image reconstruction from Cartesian data

In this tutorial we will reconstruct an MR image from Cartesian under-sampled kspace measurements.

We use the toy datasets available in pysap, more specifically a 2D brain slice and the cartesian acquisition scheme. We compare zero-order image reconstruction with Compressed sensing reconstructions (analysis vs synthesis formulation) using the FISTA algorithm for the synthesis formulation and the Condat-Vu algorithm for the analysis formulation.Sparsity will be promoted in the wavelet domain, using either Symmlet-8 (analysis and synthesis) or undecimated bi-orthogonal wavelets (analysis only).

We remind that the synthesis formulation reads (minimization in the sparsifying domain):

$$\widehat{z} = \arg\min_{z \in C^n_\Psi} \frac{1}{2}\|y - \Omega F \Psi^* z\|_2^2 + \lambda \|z\|_1$$

and 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 $n_\Psi > n$.

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

$$\widehat{x} = \arg\min_{x \in C^n} \frac{1}{2}\|y - \Omega F x\|_2^2 + \lambda \|\Psi x\|_1 \, .$$

- Author: Chaithya G R & Philippe Ciuciu
- Date: 01/06/2021
- Target: ATSI MSc students, Paris-Saclay University

[1]:
```
# Package import
from modopt.math.metrics import ssim
#from mri.numerics.fourier import FFT2
#from mri.numerics.reconstruct import sparse_rec_condatvu, sparse_rec_fista
#from mri.numerics.utils import generate_operators
#from mri.numerics.utils import convert_mask_to_locations

from mri.operators import FFT, WaveletN, WaveletUD2
from mri.operators.utils import convert_mask_to_locations
from mri.reconstructors import SingleChannelReconstructor
import pysap
from pysap.data import get_sample_data
```

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

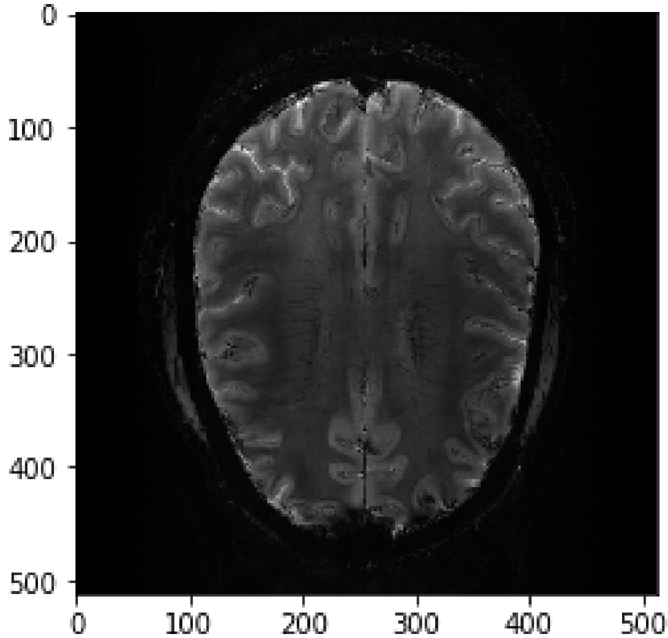
```
/home/ciuciu/anaconda3/lib/python3.7/site-
packages/mri/operators/fourier/cartesian.py:33: UserWarning: pynufft python
package has not been found. If needed use the master release. Till then you
cannot use NUFFT on GPU
  warnings.warn("pynufft python package has not been found. If needed use "
/home/ciuciu/anaconda3/lib/python3.7/site-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

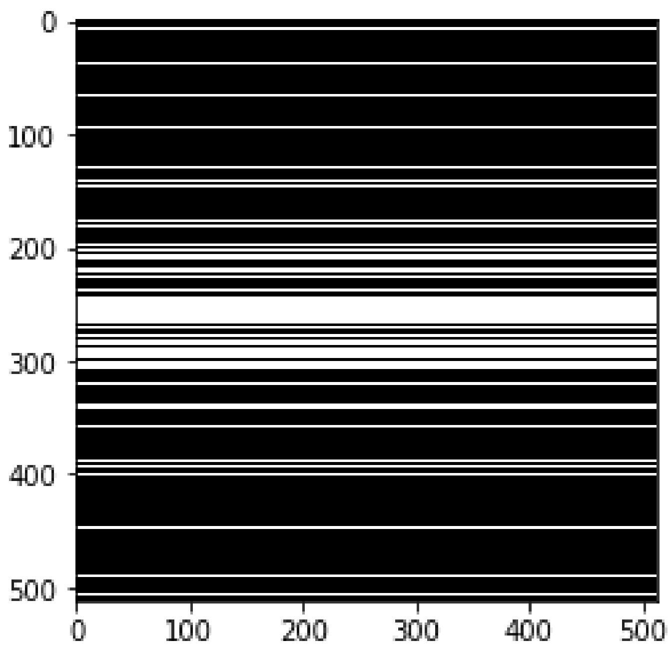
[2]:

```python
# Loading input data
image = get_sample_data('2d-mri')

# Obtain K-Space Cartesian Mask
mask = get_sample_data("cartesian-mri-mask")


# View Input
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

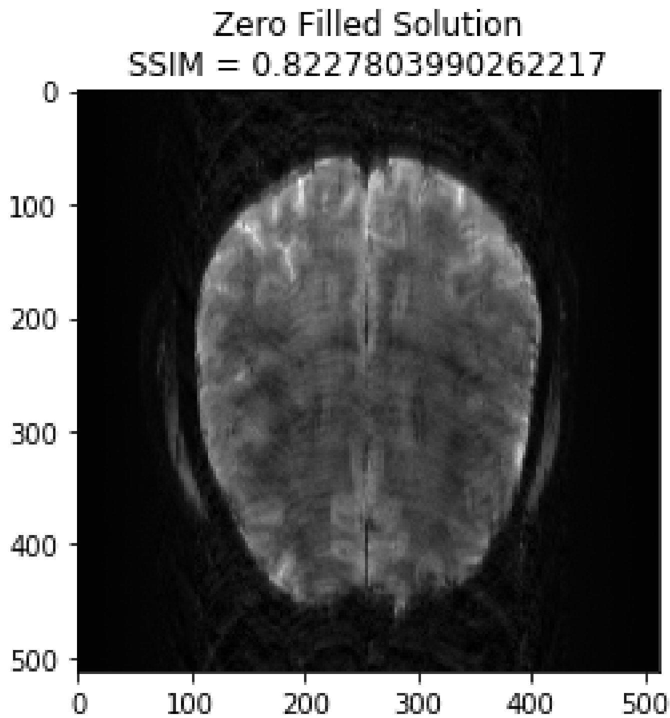
[3]:
```
#kspace_loc = convert_mask_to_locations(np.fft.fftshift(mask.data))
# Generate the subsampled kspace
#fourier_op = FFT2(samples=kspace_loc, shape=image.shape)
#kspace_data = fourier_op.op(image)

kspace_loc = convert_mask_to_locations(mask.data)
fourier_op = FFT(samples=kspace_loc, shape=image.shape)
kspace_data = fourier_op.op(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]:
```
image_rec0 = pysap.Image(data=fourier_op.adj_op(kspace_data),
                         metadata=image.metadata)
plt.imshow(np.abs(image_rec0), cmap='gray')
# Calculate SSIM
base_ssim = ssim(image_rec0, image)
plt.title('Zero Filled Solution\nSSIM = ' + str(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_scales=4)
#linear_op = WaveletN(wavelet_name="sym8", nb_scales=4,␣
 ↪padding_mode="periodization")

regularizer_op = SparseThreshold(Identity(), 2 * 1e-7, thresh_type="soft")

reconstructor = SingleChannelReconstructor(
    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 1.100000000000056
The lipschitz constraint is satisfied
```

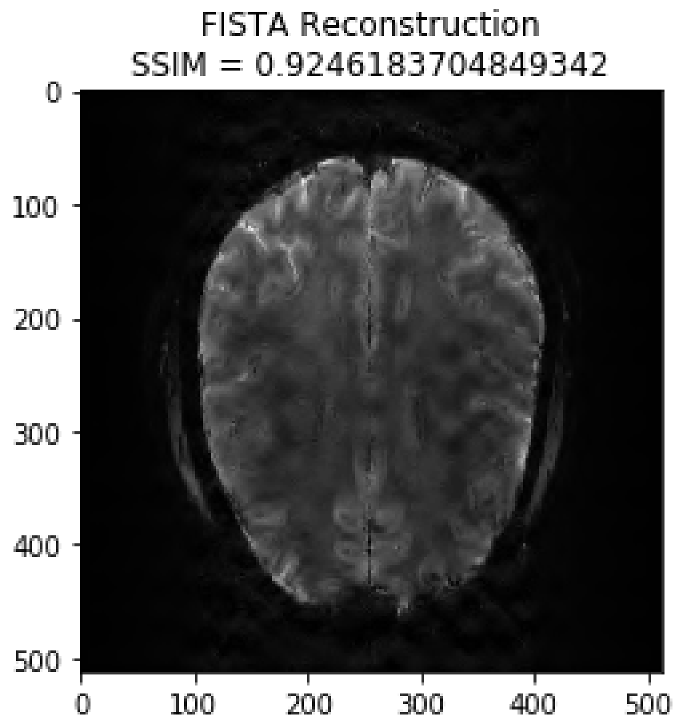
[6]:
```python
x_final, costs, metrics = reconstructor.reconstruct(
    kspace_data=kspace_data,
    optimization_alg='fista',
    num_iterations=200,
)
image_rec = pysap.Image(data=np.abs(x_final))
plt.imshow(np.abs(image_rec), cmap='gray')
recon_ssim = ssim(image_rec, image)
plt.title('FISTA Reconstruction\nSSIM = ' + str(recon_ssim))
plt.show()
```

```
WARNING: Making input data immutable.
N/A% (0 of 200) |                        | Elapsed Time: 0:00:00 ETA:  --:--:--

 - mu:  2e-07
 - lipschitz constant:  1.100000000000056
 - data:  (512, 512)
 - wavelet:  <mri.operators.linear.wavelet.WaveletN object at 0x7f88cfee07d0> -
4
 - max iterations:  200
 - image variable shape:  (512, 512)
 - alpha variable shape:  (291721,)
----------------------------------------
Starting optimization…

100% (200 of 200) |######################| Elapsed Time: 0:00:15 Time:  0:00:15

 - final iteration number:  200
 - final log10 cost value:  6.0
 - converged:  False
Done.
Execution time:  15.797152886050753  seconds
----------------------------------------
```



1.3 POGM optimization

[7]:
```
x_final, costs, metrics = reconstructor.reconstruct(
    kspace_data=kspace_data,
    optimization_alg='pogm',
    num_iterations=200,
)
image_rec = pysap.Image(data=np.abs(x_final))
plt.imshow(np.abs(image_rec), cmap='gray')
recon_ssim = ssim(image_rec, image)
plt.title('POGM Reconstruction\nSSIM = ' + str(recon_ssim))
plt.show()
```

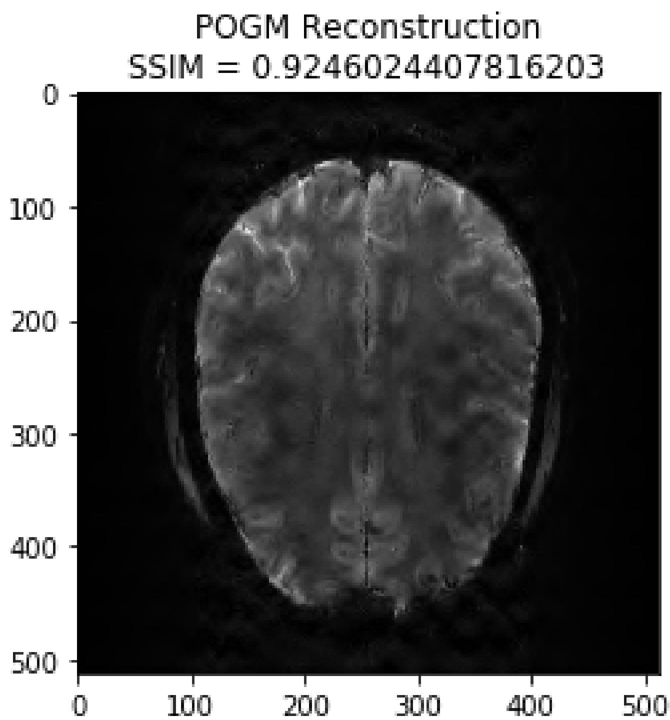
```
N/A% (0 of 200) |                        | Elapsed Time: 0:00:00 ETA:  --:--:--

 - mu:  2e-07
 - lipschitz constant:  1.100000000000056
 - data:  (512, 512)
 - wavelet:  <mri.operators.linear.wavelet.WaveletN object at 0x7f88cfee07d0> -
4
 - max iterations:  200
 - image variable shape:  (1, 512, 512)
```

```
----------------------------------------
Starting optimization…

100% (200 of 200) |######################| Elapsed Time: 0:00:18 Time:  0:00:18

 - final iteration number:  200
 - final log10 cost value:  6.0
 - converged:  False
Done.
Execution time:  18.982542266021483  seconds
----------------------------------------
```



1.4 Analysis formulation: Condat-Vu reconstruction

[8]:
```
linear_op = WaveletUD2(
    wavelet_id=24,
    nb_scale=4,
)
```

[9]:
```
reconstructor = SingleChannelReconstructor(
    fourier_op=fourier_op,
    linear_op=linear_op,
    regularizer_op=regularizer_op,
```

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

```
WARNING: Making input data immutable.
```

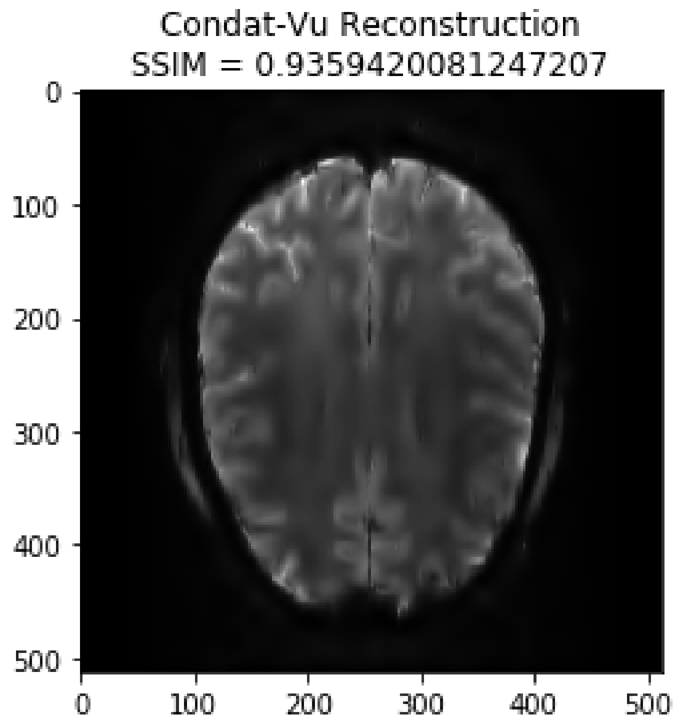
```
Lipschitz constant is 1.1
The lipschitz constraint is satisfied
```

[10]:
```
x_final, costs, metrics = reconstructor.reconstruct(
    kspace_data=kspace_data,
    optimization_alg='condatvu',
    num_iterations=200,
)
image_rec = pysap.Image(data=np.abs(x_final))
plt.imshow(np.abs(image_rec), cmap='gray')
recon_ssim = ssim(image_rec, image)
plt.title('Condat-Vu Reconstruction\nSSIM = ' + str(recon_ssim))
plt.show()
```

```
 - mu:  2e-07
 - lipschitz constant:  1.1
 - tau:  0.937465492611657
 - sigma:  0.5
 - rho:  1.0
 - std:  None
 - 1/tau - sigma||L||^2 >= beta/2:  True
 - data:  (512, 512)
 - wavelet:  <mri.operators.linear.wavelet.WaveletUD2 object at 0x7f88d17aefd0>
- 4
 - max iterations:  200
 - number of reweights:  0
 - primal variable shape:  (512, 512)
 - dual variable shape:  (2621440,)
----------------------------------------
Starting optimization…
```

```
100% (200 of 200) |######################| Elapsed Time: 0:03:16 Time:  0:03:16
```

```
 - final iteration number:  200
 - final cost value:  1000000.0
 - converged:  False
Done.
Execution time:  198.91456897603348  seconds
----------------------------------------
```


Condat-Vu Reconstruction
SSIM = 0.9359420081247207
0
100
200
300
400
500
0
100
200
300
400
500