EEE 475/575 Medical Image Reconstruction & Processing
Homework 5
25 May 2020, Monday at 23:59

GUIDELINES FOR HOMEWORK SUBMISSION

Instructions:

1. NO submission via E-MAIL (all emails will be discarded).
2. NO submission of RAR or ZIP files (rar or zip files will be discarded).
3. You should upload your solutions to Moodle as two separate files. One PDF file (should be named name_surname_id_hw#.pdf), and one MATLAB m file (should be named name_surname_id_hw#.m).
4. Submission system will remain open for 1 day after the deadline. No points will be lost if you submit your assignment within 12 hours of the deadline. There will be a 50% penalty if you submit after 12 hours but within 24 hours past the deadline. No submissions beyond 24 hours past the deadline.

MATLAB File Guidelines

1. It should be a single m file containing the codes for all questions (if you upload many m files, we will not evaluate any of them).
2. There is a template on Moodle to help you organize your solution in one m file, you should use it in all HWs.
3. Read the guidelines in the template file and follow each and every step.
4. If you do not upload your m file, your homework will not be evaluated (i.e., 0 pts). There will not be any exceptions to this rule.
5. If your m file gives runtime errors, your homework will not be evaluated (i.e., 0 pts). There will not be any exceptions to this rule.

PDF Report Guidelines

1. You get your points mainly from the PDF report file. The report should be typeset (no handwriting allowed). The PDF file should contain all results and plots. Unclear presentation of results will be penalized heavily. No partial credits to unjustified answers.
2. Maximum file size that can be uploaded to Turnitin is 40 MB. Pay attention to this before submitting your file.
3. You should properly name your pdf file and upload it to Turnitin upload area. We will discard all improperly named submissions (i.e., 0 pts).
4. After each question or each part you should properly display your MATLAB results, plots and MATLAB codes for that part (zero points for missing outputs).

PREPARATION

From the Moodle page of the class, download the following files:

- **multicoil-random.mat:** Contains fully-sampled 2DFT data acquired using an 8-channel coil, together with random sampling masks. im(Nx,Ny,Nc) contains the fully-sampled images for each channel, mask(Nx,Ny) contains a random sampling mask with 32x32 central calibration region. Here, Nx = 256 and Ny = 256 are image dimensions, and Nc = 8 is the number of coils.

- **@Wavelet.zip:** This file contains @Wavelet folder in it. Keep this folder in your working directory. You can generate a wavelet operator, perform forward and inverse wavelet transforms as follow:

```
>> wv = Wavelet('Daubechies',4,4); % generate wavelet operator
>> coeffW = wv*im; % take forward wavelet transform
>> im2 = wv'*coeffW; % take inverse wavelet transform. Note that we use the
adjoint wv (i.e., wv'), which is essentially the inverse operator.
```

Note 1: Throughout this homework, use the "montage" function whenever you need to display images from all coils in a compact form.

Note 2: Throughout this homework, normalize the images before using PSNR and SSIM.

l_1-SPIRiT

This homework will be concerned with the implementation of a ***l_1*-SPIRiT reconstruction** for compressed sensing reconstruction of randomly undersampled Cartesian MRI data sets.

1) Display the Data: Display the magnitude images for all coils, and the corresponding k-space spectrums. Perform a sum-of-squares (SoS) combination of coils and display the final image. We will use this image as the *reference image* in the rest of the homework.

2) Sampling Mask and Random Undersampled Images: Display the sampling mask as an image. This mask is a "logical" matrix made of ones and zeros, where ones correspond to the locations that will be sampled during MRI acquisition. Determine the overall acceleration corresponding to the given mask, where $R_{overall}$ = 1/(ratio of k-space acquired).

Use the mask to generate the random undersampled MRI images. Here, a simple element-wise multiplication of the mask with the k-space data is sufficient to generate the undersampled k-space data for each coil. Display the resulting k-space spectrums and the images (magnitude of the zero-fill image) for all coils using the "montage" function. Perform a SoS combination of coils and display the final zero-fill image. Compute PSNR and SSIM.

Comment on how these undersampled images look when compared to uniformly undersampled images (i.e., skipping lines in k-space).

3) SPIRiT Kernel Weights: Here, we will only consider the case for a 3x3 kernel for simplicity. Write a function that takes the central 32x32 calibration data in k-space to compute SPIRiT kernel weights, kernel =calibrateSpirit(data_calib,λ). Here, data_calib is the k-space data extracted from the calibration region, λ is the regularization parameter used in kernel estimation stage of SPIRiT.

For 3x3 kernel size, we are considering all 8 acquired neighborhood samples for the chosen coil, and all 9 acquired samples from the other coils. For that case, "kernel" is a 2D matrix of size (9×Nc-1) × Nc.

Display the kernel as an image (magnitude only) for λ=0.

4) SPIRiT Reconstruction: Write a MATLAB function that takes undersampled k-space data, and computes one iteration of SPIRiT reconstruction, imr = spirit(datau, kernel), where datau is the 2D FFT of the randomly undersampled image imu.

Select a suitable regularization weight λ (by trial and error). Display the corresponding kernel as an image (magnitude only).

Reconstruct and display the magnitude images from all coils (for one iteration only). Perform a sum-of-squares (SoS) combination of coils and display the final image. Also display the error image (i.e., the magnitude of the difference between the final image and the reconstructed image). Compute PSNR and SSIM. Comment on the results.

5) l_1 Regularization in Wavelet Domain: Write a MATLAB function that takes an image and returns a version that is l_1 regularized in wavelet domain (i.e., sparsity is enforced in wavelet domain): imth = l1wavelet(imr,beta), where imr is an input 2D image and β is the threshold in wavelet domain. This function will compute the wavelet transform of the image imr, apply the following soft thresholding on each wavelet coefficient, and take inverse wavelet transform:

$$S_\beta(x) = \frac{x}{|x|}\max(0, |x| - \beta)$$

Apply wavelet regularization on the images from Question 2 (i.e., random undersampled images from each coil). Note that you will apply wavelet regularization on complex-valued images. The threshold β may be chosen relative to the maximum coefficient. As an example, the coefficients that are smaller than 1/500 of the maximum coefficient may be set to zero.

First, display the magnitude of the wavelet transform for images from all coils from Question 2, before thresholding (**Note:** Display them in logarithmic scale, like we do for k-space spectrum). Next, do the following for a 3 different β values (choose the values so that you can clearly see the effect of β): Display the magnitude images from all coils after thresholding. Display the final SoS-combined image and the error image. Compute PSNR and SSIM. Comment on the results.

6) Iterative l_1-SPIRiT Reconstruction: Now, we will apply SPIRiT and l_1 regularization iteratively, with an added data consistency step in between. Write a function imr = l1spirit(datau, kernel, beta, numiter) that performs the following steps for "numiter" iterations:

Here are the steps for each iteration:

1. Apply SPIRiT to reconstruct full k-space images for each coil.
2. Apply l_1 regularization in wavelet domain to each image. Choose a reasonable β (by trial and error), and keep it constant across iterations.
3. Enforce data consistency in k-space by replacing the reconstructed k-space data with the acquired data at sampled k-space locations.

Here, you will use the functions you have written in Questions 4 and 5, with the added data consistency step.

Perform 10-20 iterations of these steps. At each iteration, display the final SoS-combined image and the error image. Plot PSNR and SSIM as a function of iteration number. Comment on the results.