

Computational MR imaging

Laboratory 4: Reconstruction of non-Cartesian k-space data

Code submission is due by Sunday 23:55 the week of Tuesday session. Please upload your code to StudOn in a described format. Late submissions will not be accepted.

Please include the lab04.py and utils.py in your submission without changing their names.

Learning objectives

- Reconstruct non-Cartesian MRI data using gridding and NUFFT toolbox
- Apply gridding operations: density compensation, oversampling and deapodization
- Learn to use the NUFFT toolbox

1. Radial sampling pattern:

- a. Data load
 - i. Radial kdata is loaded on *k_radial*.
 1. Column: the readout dimension for each radial line.
 2. Radial acquisition angle: 111.246117975° (golden angle)
 - ii. Plot the acquired k-space.
- b. Generate a sampling trajectory that corresponds to this data for the reconstruction. **Figure 1** shows a plot of the first 10 spokes of such a trajectory for reference.
 - i. Implement a method, *get_traj*.
 1. The spoke length should be 1 from -0.5 to 0.5.
 2. The first spoke should start at the angle of 90° .
 3. Use *self.PI* and *self.GA* for π and golden angle, respectively. They are in *radian* unit.
 4. Think about a complex number in polar form.
- c. Implement a method, *calc_nyquist*, to calculate the nyquist rate.
 - i. Consider what does each axis of *k_radial* mean.

k-space trajectory: 10 spokes

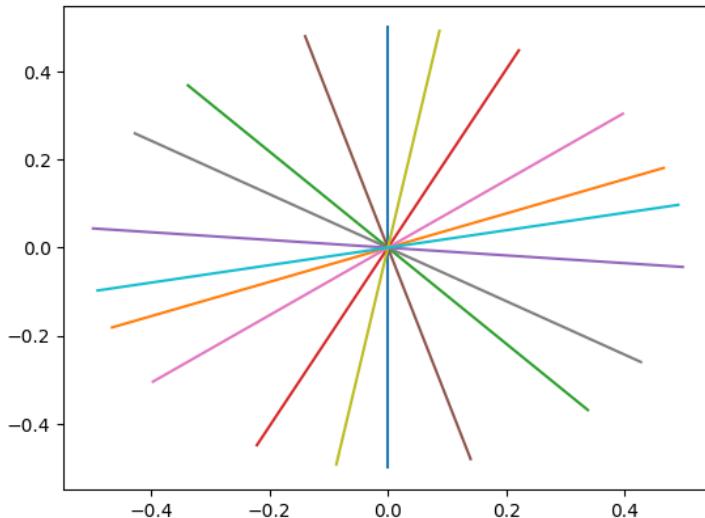


Figure 1: Radial trajectory with golden angle increment. Note that the first angle is at $\pi/2$.

2. Basic gridding reconstruction:

- a. Reconstruct this dataset using the provided grid function that grids 2D non-Cartesian k-space data to Cartesian k-space data using **triangular gridding kernel** of width 2
 - i. Implement a method, *grid_radial*.
 - ii. Map the radial kdata to the Cartesian space using the *grid_radial* method.
 - iii. Reconstruct the Cartesian kdata
- b. Plot the gridded k-space data and the reconstructed image.
- c. Comment on the artifacts. Can you guess what organ was imaged?

3. Density compensation:

- a. Implement a method, *get_ramp*.
 - i. Consider the minimum and maximum value of each spoke.
- b. Implement a method, *grid_radial_ds*.
 - i. Apply the ramp filter from *get_ramp* to the *k_radial* data.
 - ii. Apply gridding function to the density compensated kdata.
- c. Reconstruct density compensated Cartesian kdata.
- d. Plot gridded kspace and the reconstructed image.
- e. Do you need to employ oversampling and de-apodization on this dataset? Explain your answer.

4. Oversampling

- a. Implement a method, *grid_radial_ds_os*
 - i. Apply the ramp filter from *get_ramp* to the *k_radial* data.
 - ii. Apply gridding function to the density compensated kdata.
 1. Think about the matrix size to oversample the kdata for the gridding function.
- b. Implement a method, *center_crop_2d*.
- c. Reconstruct an oversampled data and crop it by following steps below.
 - i. Oversample the density compensated kdata.
 - ii. Apply inverse FFT.
 - iii. Crop the reconstructed image.
- d. Plot the gridded kspace data and both reconstructed images, oversampled and cropped.

5. De-apodization

- a. Compute the de-apodization function in the image domain and apply to the gridded image with oversampling of 2.
 - i. Implement a method, *get_c*, to get a gridding kernel in the image domain.
 1. think of the convolution of the delta function.
 2. The gridding kernel should be in the image domain.
 - ii. Implement a method, *deapodization*.
 1. Get the gridding kernel using *get_c*.
 2. Reconstruct the density compensated oversampled kdata

- a. Apple deapodization to the reconstructed image.
- 3. Crop the oversampled image to its original shape.

$$\hat{m}(x, y) = \frac{\mathbf{1}}{c(x, y) + a} \left\{ [(m(x, y) * s(x, y))c(x, y)] * \text{III} \left(\frac{x}{FOV_x}, \frac{y}{FOV_y} \right) \right\}$$

6. NUFFT toolbox

See these links for NUFFT toolbox: [Source](#), [Tutorial](#)

- a. Reconstruct the radial dataset using a widely used NUFFT toolbox from the research community.
 - i. Implement a method, *nufft_traj*
 1. Keep in mind that the length of the trajectory spoke is 2π ($-\pi \sim \pi$) and the first spoke starts at the angle of 90° .
 2. Trajectory for the NUFFT operator needs to be in shape of (2, x) where 2 is the separate channels of real and imaginary part of the trajectory.
 - ii. Implement a method, *nufft_kdata*
 1. Use density compensated kdata
 - iii. Implement a method, *nufft_recon*
 1. Call *nufft_traj* and *nufft_kdata* to define trajectory and kdata for NUFFT.
 2. Define a NUFFT adjoint operator to reconstruct the kdata.
 3. Reconstruct the kdata with the NUFFT adjoint operator.
- b. Compare NUFFT reconstructions with gridding reconstruction using the triangular kernel and discuss results.