Reconstruction Methods for Magnetic Resonance Imaging

Martin Uecker
Introduction

Topics:

- Image Reconstruction as Inverse Problem
- Parallel Imaging
- Non-Cartestian MRI
- Subspace Methods
- Model-based Reconstruction
- Compressed Sensing
Tentative Syllabus

- 01: Apr 12 Introduction
- 02: Apr 19 Parallel Imaging as Inverse Problem
- 03: Apr 26 –
- 04: May 03 Iterative Reconstruction Algorithms
- 05: May 10 Non-Cartesian MRI
- 06: May 17 Nonlinear Inverse Reconstruction
- 07: May 24 Reconstruction in k-space
- 08: May 31 Reconstruction in k-space
- 09: Jun 07 Subspace methods
- 10: Jun 14 Subspace methods - ESPIRiT
- 11: Jun 21 Model-based Reconstruction
- 12: Jun 28 Model-based Reconstruction
- 13: Jul 05 Compressed Sensing
- 14: Jul 12 Compressed Sensing
Today

- Review of last lecture
- Noise Propagation
- Iterative Reconstruction Algorithms
- Software
Phased Array

Signal is Fourier transform of magnetization image $m$ weighted by coil sensitivities $c_j$: 

$$s_j(t) = \int d\vec{x} \rho(\vec{x}) c_j(\vec{x}) e^{-i2\pi \vec{k}(t) \cdot \vec{x}}$$

Images of a human brain from an eight channel array:
Channel Combination

RSS

MVUE

MMSE
Parallel MRI

**Goal:** Reduction of measurement time

- Subsampling of k-space
- Simultaneous acquisition with multiple receive coils

- Coil sensitivities provide spatial information
- Compensation for missing k-space data

Parallel MRI: Undersampling

Undersampling

$k_{phase}$

$k_{read}$

$k_{partition}$

Aliasing

$k_{phase}$

$k_{read}$
Parallel Imaging as Inverse Problem

**Model:** Signal from multiple coils (image $\rho$, sensitivities $c_j$):

$$s_j(t) = \int_{\Omega} d\vec{x} \rho(\vec{x}) c_j(\vec{x}) e^{-i2\pi\vec{x} \cdot \vec{k}(t)} + n_j(t)$$

**Assumptions:**

- Image is square-integrable function $\rho \in L^2(\Omega, \mathbb{C})$
- Additive multi-variate Gaussian white noise $n$

**Problem:** Find best approximate/regularized solution in $L^2(\Omega, \mathbb{C})$. 
Discretization of Linear Inverse Problems

**Continuous** integral operator \( F : f \mapsto g \) with kernel \( K \):

\[
g(t) = \int_{a}^{b} ds \, K(t, s)f(s)
\]

**Discrete** system of linear equations:

\[
y = Ax
\]

**Considerations:**

- Discretization error
- Efficient computation
- Implicit regularization

<table>
<thead>
<tr>
<th></th>
<th>continuous</th>
<th>discrete</th>
</tr>
</thead>
<tbody>
<tr>
<td>operator</td>
<td>( F )</td>
<td>( A )</td>
</tr>
<tr>
<td>unknown</td>
<td>( f )</td>
<td>( x )</td>
</tr>
<tr>
<td>data</td>
<td>( g )</td>
<td>( y )</td>
</tr>
</tbody>
</table>
Discretization for Parallel Imaging

**Discrete Fourier basis:**

\[ f(x, y) \approx \sum_{l=-N}^{N} \sum_{k=-N}^{N} \hat{a}_{l, k} e^{i2\pi \left( \frac{lx}{\text{FOV}_x} + \frac{ly}{\text{FOV}_y} \right)} \]

- Efficient computation (FFT)
- Approximates \( R(F^H) \) extremely well (for smooth \( c_j \))
- Voxels: Dirichlet kernel \( D_N(\frac{x}{\text{FOV}_x})D_N(\frac{y}{\text{FOV}_y}) \)
Discretization
SENSE: Discretization

Weak voxel condition: Images from discretized subspace should be recovered exactly (from noiseless data).

Common choice:

\[ f(x, y) \approx \sum_{r,s} \delta(x - r \frac{FOV_x}{N_x})\delta(y - s \frac{FOV_y}{N_y}) \]

- Efficient computation using FFT algorithm
- Periodic sampling (⇒ decoupling)

Problem: Periodically extended k-space.
⇒ Error at the k-space boundary!
SENSE: Decoupling

System of **decoupled** equations:

\[
\begin{pmatrix}
  m_1(x, y) \\
  \vdots \\
  m_n(x, y)
\end{pmatrix}
= 
\begin{pmatrix}
  c_1(x, y_1) & c_1(x, y_2) \\
  \vdots & \vdots \\
  c_n(x, y_1) & c_n(x, y_2)
\end{pmatrix}
\cdot 
\begin{pmatrix}
  \rho(x, y_1) \\
  \rho(x, y_2)
\end{pmatrix}
\]
Discretization: Summary

- Continuous reconstruction (from finite data) is ill-posed!
- Discretization error
- Implicit regularization
  (discretized problem might be well-conditioned)

**Attention:** Be careful when simulating data! Same discretization for simulation and reconstruction
⇒ misleading results (inverse crime)
Today

- Review of last lecture
- Noise Propagation
- Iterative Reconstruction Algorithms
- Software
Complex Gaussian Distribution

Random variable $Z = X + iY$

Proper complex Gaussian:

$$Z \sim \mathcal{CN}(\mu, \sigma^2) \quad p(Z) = \frac{1}{\sigma \pi} e^{-\frac{|Z-\mu|^2}{\sigma^2}}$$

mean: $\mu = E[Z]$

variance: $\sigma^2 = E[(Z - \mu)(Z - \mu)^*]$

proper: pseudo-variance

$E[(Z - \mu)(Z - \mu)] = 0$
Multi-Variate Complex Gaussian Distribution

Random vector $Z = X + iY$

Multi-variate proper complex Gaussian distribution:

$Z \sim CN(\mu, \Sigma)$

mean $\mu = E[Z]$

covariance $\Sigma = Cov[Z, Z] = E[(Z - \mu)(Z - \mu)^H]$

pseudo-covariance $E[(Z - \mu)(Z - \mu)^T] = 0$
Linear reconstruction: $\hat{Z} = FZ$

$\hat{Z} \sim CN(F\mu, F\Sigma F^H)$

Full covariance matrix for all pixels: $F\Sigma F^H$

$\Rightarrow$ Not (always) practical
(2D size $\propto 10^9$, 3D size: $\propto 10^{14}$)
Geometry Factor

Quantity of interest: noise variance of pixel values:

\[ \sigma(x_i) = \sqrt{(F\Sigma F^H)_{ii}} \]

Spatially dependent noise:

\[ \sigma_{und.}(x) = g(x)\sqrt{R}\sigma_{full}(x) \]

Acceleration: \( R \), **Geometry factor**: \( g \)

Practical estimation: Monte-Carlo method

\[ \hat{\sigma}^2(x) = N^{-1} \sum_j |F_n_j|^2 \]

Gaussian white noise \( n_j \).
Noise Amplification in Parallel Imaging

Local noise amplification dependent on:
- Sampling pattern
- **Coil sensitivities**

Parallel MRI: Regularization

- General problem: bad condition
- Noise amplification during image reconstruction
- $L^2$ regularization (Tikhonov):

$$\arg\min_x \|Ax - y\|_2^2 + \alpha \|x\|_2^2 \Leftrightarrow (A^HA + \alpha I)x = A^Hy$$

- Influence of the regularization parameter $\alpha$:
Noise vs. Bias
Parallel MRI: Nonlinear Regularization

- Good noise suppression
- Edge-preserving

⇒ Sparsity, nonlinear regularization

\[
\arg\min_x \|Ax - y\|_2^2 + \alpha R(x)
\]

Regularization: \( R(x) = TV(x), \, R(x) = \|Wx\|_1, \ldots \)

Parallel MRI: Nonlinear Regularization

- $L^2$, $L^1$-wavelet and TV-regularization
- 3D-FLASH, 12-channel head coil
- 2D-reduction factor of 12 (phantom) and 6 (brain)
Correlated Noise - Whitening

**Cholesky decomposition:**

\[ \Sigma = LL^H \]

Lower triangular matrix \( L \):
Transform with \( W = L^{-1} \) to uncorrelated and equalized noise:

\[ CN(0, \Sigma) \implies CN(0, W\Sigma W^H) = CN(0, I) \]

Reconstruction problem: \( Ax = y \implies WAx = Wy \) Normal equations:

\[ A^H W^H WAx = A^H W^H Wy \iff \arg\min_x \| WAx - Wy \|_2 \]

**In MRI:** Noise correlations between receive channels. Whitening \( \implies \) uncorrelated virtual channels: \( Wy \)
Today

- Review of last lecture
- Noise Propagation
- Iterative Reconstruction Algorithms
- Software
Parallel MRI: Iterative Algorithms

Signal equation:

\[ s_i(k) = \int_V d\vec{x} \rho(\vec{x}) c_i(\vec{x}) e^{-i2\pi\vec{x} \cdot \vec{k}} \]

encoding functions

Discretization:

\[ A = P_k F C \]

\( A \) has size \( 256^2 \times (8 \times 256^2) \)

⇒ Iterative methods built from matrix-vector products \( A x, A^H y \)
Landweber Iteration

Gradient descent:

\[ \phi(x) = \frac{1}{2} \|Ax - y\|_2^2 \]
\[ \nabla \phi(x) = A^H(Ax - y) \]

Iteration rule:

\[ x_{n+1} = x_n - \mu A^H(Ax_n - y) \quad \text{with} \quad \mu \|A^HA\| \leq 1 \]
\[ = (I - \mu A^HA)x_n - A^Hy \]

Explicit formula for \( x_0 = 0 \):

\[ x_n = \sum_{j=0}^{n-1} (I - \mu A^HA)^j \mu A^Hy \]
Landweber Iteration

reconstruction of right singular-vectors \( (\mu = 0.9) \)

\[
x_n = \sum_{j=0}^{n-1} (I - \mu A^H A)^j \mu A^H A x = \sum_l \left( 1 - (1 - \mu \sigma_l^2) \right)^n V_l V_l^H x
\]

geometric series: \( \sum_{j=0}^{n-1} x^j = \frac{1-x^n}{1-x} \)  

SVD: \( A = \sum_j \sigma_j U_j V_j^H \)
Projection Onto Convex Sets (POCS)

Iterative projection onto convex sets:

\[ y_{n+1} = P_B P_A y_n \]

Problems: slow, not stable, \( A \cap B = \emptyset \)
Projection Onto Convex Sets (POCS)

Iterative projection onto convex sets:

\[ y_{n+1} = P_B P_A y_n \]

Problems: slow, not stable, \( A \cap B = \emptyset \)
POCSENSE

Iteration:

\[ \hat{y}_{n+1} = P_C P_y \hat{y}_n \]

(multi-coil k-space \( \hat{y} \))

Projection onto sensitivities:

\[ P_C = \mathcal{F} C C^H \mathcal{F}^{-1} \]

(normalized sensitivities)

Projection onto data \( y \):

\[ P_y \hat{y} = M y + (1 - M) \hat{y} \]

POCSENSE

Fully-sampled data should be consistent with sensitivities:

\[ P_C y_{\text{full}} = y_{\text{full}} \Rightarrow \mathcal{F} \left( C C^H - I \right) \mathcal{F}^{-1} y_{\text{full}} = 0 \]

But is not:

\[ \mathcal{F}^{-1} y_{\text{full}} \]

\[ C \]

\[ \left( C C^H - I \right) \mathcal{F}^{-1} y_{\text{full}} \]

Note: Can be used to validate coil sensitivities.
For parallel MRI: POCS corresponds to Landweber for $\mu = 1$ and normalized sensitivities.

\[
y_{n+1} = P_C P_y y_n = \mathcal{F} C \underbrace{C^H \mathcal{F}^{-1}((I - P_k)y_n + y_0)}_{x_n}
\]

Rewrite:

\[
x_{n+1} = C^H \mathcal{F}^{-1}((I - P_k)\mathcal{F} C x_n + y_0) \quad \text{with} \quad y_n = \mathcal{F} C x_n
\]
\[
= C^H C x_n - C^H \mathcal{F}^{-1} P_k \mathcal{F} C x_n + C^H \mathcal{F}^{-1} P_k y_0 \quad \text{with} \quad P_k y_0 = y_0
\]
\[
= x_n - A^H (Ax_n - y_0) \quad \text{with} \quad C^H C = I, \quad A = P_k \mathcal{F} C
\]
Krylov Subspace Methods

Krylov subspace:

\[ \mathcal{K}_n = \text{span}_{i=0}^{n} \{ T^n b \} \]

⇒ Repeated application of \( T \).

Landweber, Arnoldi, Lanczos, Conjugate gradients, ...
Conjugate Gradients

$T$ symmetric (or Hermitian)

Initialization:

$$r_0 = b - Tx_0$$
$$d_0 = r_0$$

Iteration:

$$q_i \leftarrow Td_i$$
$$\alpha = \frac{|r_i|^2}{\mathcal{R} r_i^H q_i}$$
$$x_{i+1} \leftarrow x_i + \alpha d_i$$
$$r_{i+1} \leftarrow r_i - \alpha q_i$$
$$\beta = \frac{|r_{i+1}|^2}{|r_i|^2}$$
$$d_{i+1} \leftarrow r_{i+1} + \beta d_i$$
Conjugate Gradients

$T$ symmetric (or Hermitian)

Initialization:

\[ r_0 = b - Tx_0 \]
\[ d_0 = r_0 \]

Iteration:

\[ q_i \leftarrow Td_i \]

\[ x_{i+1} \leftarrow x_i + \alpha d_i \]
\[ r_{i+1} \leftarrow r_i - \alpha q_i \]

\[ d_{i+1} \leftarrow r_{i+1} + \beta d_i \]
Conjugate Gradients

\( T \) symmetric (or Hermitian)

Initialization:
\[
\begin{align*}
    r_0 &= b - Tx_0 \\
    d_0 &= r_0
\end{align*}
\]

Iteration:
\[
\begin{align*}
    q_i &\leftarrow Td_i \\
    \alpha &= \frac{|r_i|^2}{\Re r_i^H q_i} \\
    x_{i+1} &\leftarrow x_i + \alpha d_i \\
    r_{i+1} &\leftarrow r_i - \alpha q_i \\
    \beta &= \frac{|r_{i+1}|^2}{|r_i|^2} \\
    d_{i+1} &\leftarrow r_{i+1} + \beta d_i
\end{align*}
\]
Conjugate Gradients

$T$ symmetric or Hermitian

Directions are conjugate basis:

$$p_i^H T p_j = \delta_{ij}$$

Coefficients can be directly computed:

$$b = T x = T \sum_i \alpha_i p_i = \sum_i \alpha_i T p_i$$

$$p_j^H b = \sum_i \alpha_i p_j^H T p_i = \alpha_j p_j^H T p_j$$

$$\Rightarrow \alpha_j = \frac{p_j^H b}{p_j^H T p_j}$$
Conjugate Gradients on Normal Equations

Normal equations:

\[ A^H Ax = A^H y \quad \iff \quad x = \arg\min_x \|Ax - y\|^2_2 \]

Tikhonov:

\[ (A^H A + \alpha I) x = A^H y \quad \iff \quad x = \arg\min_x \|Ax - y\|^2_2 + \alpha \|x\|^2_2 \]

CGNE:

\[ Tx = b \]

with:

\[ T = A^H A \]

\[ b = A^H y \]
Implementation

FFT
IFFT
P
FFT
IFFT
P
distribute image
point-wise multiplication with sensitivities
multi-dimensional FFT
sampling mask or projection onto data
inverse FFT
point-wise mult. with conjugate sensitivities
sum channels
repeat
data flow
Conjugate Gradient Algorithm vs Landweber

- Conjugate Gradients
- Landweber
Conjugate Gradient Algorithm vs Landweber

- Conjugate gradients
- Landweber
Conjugate Gradient Algorithm vs Landweber
Conjugate Gradient Algorithm vs Landweber

conjugate gradients

Landweber
Conjugate Gradient Algorithm vs Landweber

- Conjugate gradients
- Landweber
Conjugate Gradient Algorithm vs Landweber
Conjugate Gradient Algorithm vs Landweber

- Conjugate gradients
- Landweber

The diagram compares the performance of the Conjugate Gradient Algorithm and the Landweber method, showing the iteration paths and convergence points.
Conjugate Gradient Algorithm vs Landweber

- **Conjugate gradients**
- **Landweber**
Conjugate Gradient Algorithm vs Landweber

- Conjugate gradients
- Landweber
Conjugate Gradient Algorithm vs Landweber

Size: $41400 \times 41400$ (complex)
Nonlinear Methods

Non-linear forward problems:

- Non-linear Conjugate Gradient
- Landweber $x_{n+1} = x_n + \mu D F^H (y - F x)$
- Iteratively Regularized Gauss-Newton Method
- ...

Non-quadratic regularization:

- IST, FISTA
- ADMM
- ...

Project 1: Iterative SENSE

**Project:** Implement and study Cartesian iterative SENSE

- **Tools:** Matlab, reconstruction toolbox, python, ...
- **See website for data and instructions.**
Project 1: Iterative SENSE

Step 1: Implement Model

\[ A = P \mathcal{F} S \]
\[ A^H = S^H \mathcal{F}^{-1} P^H \]

Hints:
- Use unitary and centered (fftshift) FFT \( \| \mathcal{F} x \|_2 = \| x \|_2 \)
- Implement undersampling as a mask, store data with zero
- Check \( \langle x, A y \rangle = \langle A^H x, y \rangle \) for random vectors \( x, y \)
Project 1: Iterative SENSE

Step 2: Implement Reconstruction

Landweber (gradient descent)\textsuperscript{1}

\[ x_{n+1} = x_n + \alpha A^H(y - Ax_n) \]

Conjugate gradient algorithm\textsuperscript{2}

Step 3: Experiments

- Noise, errors, and convergence speed
- Different sampling
- Regularization

Today

- Review of last lecture
- Noise Propagation
- Iterative Reconstruction Algorithms
- **Software**
Software Toolbox

- Rapid prototyping
  (similar to Matlab, octave, ...)

- Reproducible research
  (i.e. scripts to reproduce experiments)

- Robustness and clinically feasible runtime
  (C/C++, OpenMP, GPU programming)
Programming library

- Consistent API based on multi-dimensional arrays
- FFT and wavelet transform
- Generic iterative algorithms (conjugate gradients, IST, FISTA, IRGNM, ADMM, . . .)
- Transparent GPU acceleration of most functions

Command-line tools

- Simple file format
- Interoperability with Matlab
- Basic operations: fft, crop, resize, slice, . . .
- Sensitivity calibration and image reconstruction
Software

- Available for Linux and Mac OS X (64 bit)
  http://mrirecon.github.io/bart/

- Requirements: FFTW, LAPACK (CUDA, ACML)

Ubuntu:
sudo apt-get install libfftw3-dev
sudo apt-get install liblapack-dev
sudo apt-get install libpng-dev

Mac OS X:
sudo port install fftw-3-single
sudo port install gcc47
sudo port install libpng
Data Files

Data files store multi-dimensional arrays.

example.hdr  \(\Leftarrow\) Text header
example.cfl  \(\Leftarrow\) Data: complex single-precision floats

Text header:

```
# Dimensions
1 230 180 8 2 1 1 1 1 1 1 1 1 1 1 1 1
```

Matlab functions:
```
data = readcfl('example');
writecfl('example', data)
```

C Functions (using memory-mapped IO)
Rapid Prototyping

Data processing using command line tools:

```
# resize 0 320 tmp in
# fft -i 7 out tmp
```

Matlab/Octave:

```
data = squeeze(readcfl('out'));
data = bart('fft -i 7', data);
imshow3(abs(data), []);
```
#include <complex.h>

#include "num/fft.h"
#include "misc/mmio.h"

int main()
{
    int N = 16;
    long dims[N];
    complex float* in = load_cfl("in", N, dims);
    complex float* out = create_cfl("out", N, dims);

    fftc(N, dims, 1 + 2 + 4, out, in);
}
Reconstruction Algorithms

- Iterative SENSE\textsuperscript{1}
- Nonlinear inversion\textsuperscript{2}
- ESPIRiT calibration and reconstruction\textsuperscript{3}
- Regularization: L2 and L1-wavelet

pics: A Tool for Parallel Imaging Compressed Sensing

> bart pics -RA:B:C:D -R ... [-t trj] kspace sens image

- parallel imaging and compressed sensing
- non-Cartesian k-space trajectories
- multiple regularization terms
- A: different types of regularization: $\ell_2$, $\ell_1$, total variation, $\ell_1$-wavelet, (multi-scale) low-rank
- B: transforms along arbitrary dimensions (space, time, etc.)
- C: joint-thresholding along arbitrary dimensions
- D: regularization parameter

Note: Depending on the algorithm additional parameters (step size, number of iterations, etc.) must be set for optimal results.