EE290T: Advanced Reconstruction Methods for Magnetic Resonance Imaging

Martin Uecker
Introduction

Topics:

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

- 01: Jan 27 Introduction
- 02: Feb 03 Parallel Imaging as Inverse Problem
- 03: Feb 10 Iterative Reconstruction Algorithms
- 04: Feb 17 (holiday)
- 05: Feb 24 Non-Cartesian MRI
- 06: Mar 03 Nonlinear Inverse Reconstruction
- 07: Mar 10 Reconstruction in k-space
- 08: Mar 17 Reconstruction in k-space
- 09: Mar 24 (spring recess)
- 10: Mar 31 Subspace methods
- 11: Apr 07 Model-based Reconstruction
- 12: Apr 14 Compressed Sensing
- 13: Apr 21 Compressed Sensing
- 14: Apr 28 TBA
Today

- Review of last lecture
- Parallel Imaging as Inverse Problem
- Project 1: Iterative SENSE
- (Software)
Direct Image Reconstruction

- **Assumption:** Signal is Fourier transform of the image:

\[ s(t) = \int d\vec{x} \rho(\vec{x}) e^{-i2\pi \vec{x} \cdot \vec{k}(t)} \]

- Image reconstruction with inverse DFT

\[ \vec{k}(t) = \gamma \int_0^t d\tau \tilde{G}(\tau) \]

\[ \Rightarrow \text{sampling} \quad \Rightarrow \text{iDFT} \quad \Rightarrow \text{k-space} \quad \Rightarrow \text{image} \]
Requirements

- Short readout (signal equation holds for short time spans only)
- Sampling on a Cartesian grid

$\Rightarrow$ Line-by-line scanning

![Diagram of MRI sequence](image)

measurement time:

$\text{TR} \geq 2\,\text{ms}$

2D: $N \approx 256 \Rightarrow \text{seconds}$

3D: $N \approx 256 \times 256 \Rightarrow \text{minutes}$
Sampling
Dirichlet Kernel

\[ D_n(x) = \sum_{k=-n}^{n} e^{2\pi i kx} = \frac{\sin(\pi(2n+1)x)}{\sin(\pi x)} \]

\[ (D_n \ast f)(x) = \int_{-\infty}^{\infty} dy \, f(y) D_n(x - y) = \sum_{k=-n}^{n} \hat{f}(k) e^{2\pi i kx} \]

FWHM \approx 1.2

\[ D_{16} \]
Gibbs Phenomenon

- Truncation of Fourier series
  ⇒ **Ringing** at jump discontinuities

Rectangular wave and Fourier approximation
Direct Image Reconstruction

**Assumptions:**

- Signal is Fourier transform of the image
- Sampling on a Cartesian grid
- Signal from a limited (compact) field of view
- Missing high-frequency samples are small
- Noise is neglectable
Inverse Problem

Definition (Hadamard): A problem is called well-posed if

1. there exists a solution to the problem (existence),
2. there is at most one solution to the problem (uniqueness),
3. the solution depends continuously on the data (stability).

(we will later see that all three conditions can be violated)
Moore-Penrose Generalized Inverse

(A linear and bounded)

\( x \) is **least-squares solution**, if

\[
\| Ax - y \| = \inf \{ \| A\hat{x} - y \| : \hat{x} \in X \}
\]

\( x \) is **best approximate solution**, if

\[
\| x \| = \inf \{ \| \hat{x} \| : \hat{x} \text{ least-squares solution} \}
\]

Generalized inverse \( A^\dagger : D(A^\dagger) := R(A) + R(A)^\perp \mapsto X \) maps data to the best approximate solution.
Tikhonov Regularization

(inverse not continuous: \( \| A^{-1} \| = \infty \))

Regularized optimization problem:

\[ x^\delta_{\alpha} = \text{argmin}_x \| A x - y^\delta \|^2 + \alpha \| x \|^2 \]

Explicit solution:

\[ x^\delta_{\alpha} = \left( A^H A + \alpha I \right)^{-1} A^H y^\delta \]

Generalized inverse:

\[ A^\dagger = \lim_{\alpha \to 0} A^\dagger_{\alpha} \]
Tikhonov Regularization: Bias vs Noise

Noise contamination:

\[ y^\delta = y + \delta n \]
\[ = Ax + \delta n \]

Reconstruction error:

\[ x - x_\alpha^\delta = x - (A^H A + \alpha I)^{-1} A^H y^\delta \]
\[ = x - (A^H A + \alpha I)^{-1} A^H Ax + (A^H A + \alpha I)^{-1} A^H \delta n \]
\[ = \alpha(A^H A + \alpha I)^{-1} x + (A^H A + \alpha I)^{-1} A^H \delta n \]

\[ \text{approximation error} \quad \text{data noise error} \]

Morozov’s discrepancy principle:

Largest regularization with \( \|Ax_\alpha^\delta - y^\delta\|_2 \leq \tau \delta \)
Phased Array

Advantages:

▶ SNR of small surface coils
▶ FOV of large coils
▶ Parallel Imaging

Roemer, The NMR Phased Array, MRM 1990.
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:
Image Combination

What to do with all those images?

Estimation of a single image:

- Minimum-variance unbiased estimator
- Root-of-sum-of-squares
- (Minimum mean-squared error)
Minimum-Variance Unbiased Estimator

- Noise-optimal unbiased estimation:

\[ \hat{m}(x) = \sum_i \frac{c_i^*(x)m_i(x)}{\sum_i |c_i(x)|^2} \]

- Proof: Gauss-Markov theorem.

- **Coil sensitivities needed!**

(Assumption: uncorrelated noise)
Minimum-Variance Unbiased Estimator

Noise-optimal unbiased estimation:

\[ \hat{m}(x) = \sum_i \frac{c_i^*(x)m_i(x)}{\sum_i |c_i(x)|^2} \]

\[ = \left( C_x^H C_x \right)^{-1} C_x^H \begin{pmatrix} m_1(x) \\ \vdots \\ m_N(x) \end{pmatrix} \]

\[ C_x = \begin{pmatrix} c_1(x) \\ \vdots \\ c_N(x) \end{pmatrix} \]

Proof: Gauss-Markov theorem.

Coil sensitivities needed!

(Assumption: uncorrelated noise)
Minimum Mean Squared Error

- Biased estimation:

\[ \hat{m}_\alpha(x) = \sum_i \frac{c_i^*(x)m_i(x)}{\lambda + \sum_i |c_i(x)|^2} \]

- \( \lambda \) controls tradoff between bias and SNR
  (can achieve optimal MSE)

- Differs from MVUE only by (spatially variant) scaling!

- **Coil sensitivities needed!**

- Tikhonov regularization: \((C^HC + \lambda I)^{-1}C^H\)
Root of Sum of Squares

- Nonlinear estimation of magnitude image:

\[
\hat{m}(x) = \sqrt{\sum_i |m_i(x)|^2}
\]

- Basic idea: Pixel values as estimates for sensitivities:

\[
c_i(x) \approx \frac{m_i(x)}{\sqrt{\sum_i |m_i(x)|^2}}
\]

- Final image is weighted by coil sensitivities (bias):

\[
\hat{m}(x) = \sqrt{\sum_i |m_i(x)|^2} = |m(x)| \sqrt{\sum_i |c_i(x)|^2}
\]

- No coil sensitivities needed!
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

\[ k_{\text{phase}} \]

\[ k_{\text{read}} \]

Aliasing

\[ k_{\text{partition}} \]

\[ k_{\text{phase}} \]
Parallel Imaging

- SMASH\(^1\)
- SENSE, CG-SENSE\(^2\)
- GRAPPA\(^3\), ARC
- JSENSE, NLINV
- SPIRiT
- ESPIRiT
- SAKE, CLEAR
- ...

Image Domain and k-Space Domain

Convolution theorem: $\mathcal{F} \{ m \cdot c \} = \mathcal{F} \{ m \} \ast \mathcal{F} \{ c \}$
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 $L_2(\Omega, \mathbb{C})$. 
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 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>continuous operator</th>
<th>discrete unknown</th>
<th>continuous unknown</th>
<th>discrete unknown</th>
</tr>
</thead>
<tbody>
<tr>
<td>$F$</td>
<td>$f$</td>
<td>$A$</td>
<td>$x$</td>
</tr>
<tr>
<td>$g$</td>
<td>$y$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Discretization for Parallel Imaging

Forward operator $F$ with kernel:

$$K_j(\vec{k}, \vec{x}) = c_j(\vec{x}) e^{-i2\pi \vec{k} \cdot \vec{x}}$$

Finite number of discrete samples $k \in S$ and channels $j = 1, \cdots, N$ \implies $F$ already has finite rank

Expansion of image $f$ using basis functions $b_l$:

$$f(\vec{x}) = \sum_l a_l b_l(\vec{x})$$
Discretization for Parallel Imaging

\[ F \text{ finite rank} \]

Tikhonov regularization: \( A_{\alpha}^\dagger y \in R(F_{\alpha}^\dagger) \)

Finite dimensional space:

\[ R(F_{\alpha}^\dagger) = R(F^H) = \text{span}_{k,j}\{enc_{k,j}\} \]

⇒ Exact solution can be expressed in a finite basis!

Problem: \( enc_{k,j} \) are usually only known approximately
Discretization for Parallel Imaging

Discrete Fourier basis:

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

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

Slightly more elements than extend of k-space data:

\[ m \cdot c \]

\[ \mathcal{F}\{m\} \star \mathcal{F}\{c\} \]
SENSE: Classical Theory

Encoding functions:

\[ y_{jk} = \int \; d\vec{x} \; \underbrace{c_j(\vec{x})e^{-i2\pi \vec{k} \cdot \vec{x}}}_{\text{enc}_{jk}} \rho(\vec{x}) \]

\[ = \langle \text{enc}_{jk}, \rho \rangle \]

Discretization:

\[ \rho(\vec{x}) \approx \sum_l a_l b_l(\vec{x}) \]

(In SENSE, \( b_l \) are called ideal voxel functions)

Encoding matrix:

\[ E_{jk,l} = \langle \text{enc}_{jk}, b_l \rangle \]

maps coefficients to data:

\[ y_{jk} = \langle \text{enc}_{jk}, \rho \rangle \approx \sum_l \langle \text{enc}_{jk}, b_l \rangle a_l = \sum_l E_{jk,l} a_l \]
Linear reconstruction of coefficients from samples using a reconstruction matrix $R$:

$$a_l = \sum_{jk} R_{l,jk} y_{jk}$$

Encoding functions transformed to voxel functions:

$$f_l = \sum_{jk} R_{l,jk} \text{enc}_{jk}$$

Encoding functions: spatial sensitivity of samples $y_{jk} = \langle \text{enc}_{jk}, \rho \rangle$

Voxel functions: spatial sensitivity of coefficients $a_l = \langle f_l, \rho \rangle$
SENSE: Classical Theory

Still missing: basis $b_l$ and reconstruction $F$

Given: basis $b_l$ (ideal voxel functions)

- **Strong voxel condition**: Voxel functions should be close to basis elements $b_l$ in least-squares sense:

  $$R = \arg\min_{R} \sum_l \| b_l - f_l \|^2$$

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

  $$RE = I$$

  $$a_l = \sum_{jk} R_{l,jk} y_{jk} = \sum_{jk} R_{l,jk} \sum_l E_{jk,l} a_l$$
SENSE: Strong Voxel Condition

**Strong voxel condition**: Voxel functions should be close to basis elements $b_l$ in least-squares sense:

$$R = \arg\min_R \sum_l \|b_l - f_l\|^2$$

With the correlation matrix $K_{il,jk} = \langle enc_{il}, enc_{jk} \rangle$, this leads to (see Pruessmann et al.):

$$R = E^H K^{-1}$$

With forward operator $F$ and basis $B$:

$$K = FF^H \quad E = FB$$

**Projection of best approximation to discretized space**:

$$B^H F^H \left(FF^H \right)^{-1} = B^H F^\dagger$$
Weak voxel condition: Images from discretized subspace should be recovered exactly (from noiseless data):

\[ RE = I \]

Not fully defined if number of basis elements (voxels) is smaller than channels \( \times \) samples.

Best approximate solution in discretized subspace:

\[ R = E^\dagger = \left( E^H E \right)^{-1} E^H \]
\[ = \left( B^H F^H FB \right)^{-1} B^H F^H \]
Geometry

\[ \text{span } \text{enc}_{jk} \]

\[ F^\dagger y \]

\[ \text{span } b_l \]

\[ (FB)^\dagger y \]

\[ B^H F^\dagger y \]
Discretization: Implicit Regularization

- Continuous reconstruction is ill-posed!
- Discretized problem might even be well-conditioned

Attention: Be careful when simulating data! Same discretization for simulation and reconstruction $\Rightarrow$ misleading results (inverse crime)
Discretization

Sampling Pattern

Power Function

Noise Amplification

Cartesian

Poisson-Disc

Uniform Random

SENSE: Discretization

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: Discretization
System of **decoupled** equations:

\[
\begin{pmatrix}
m_1(x, y) \\
\vdots \\
m_n(x, y)
\end{pmatrix}
= 
\begin{pmatrix}
c_1(x, y_1) & c_1(x, y_2) & \cdots \\
c_2(x, y_1) & c_2(x, y_2) & \cdots \\
\vdots & \vdots & \ddots \\
c_n(x, y_1) & c_n(x, y_2) & \cdots
\end{pmatrix}
\cdot 
\begin{pmatrix}
\rho(x, y_1) \\
\rho(x, y_2)
\end{pmatrix}
\]
Noise Propagation

Random vector \( X \)

Signal and noise: multi-variate proper complex Gaussian distribution:

\[ X \sim \mathcal{CN}(\mu, \Sigma) \]

mean \( \mu = E[X] \), covariance \( \Sigma = \text{Cov}[X, X] = E[(X - \mu)(X - \mu)^H] \)

Linear transformation: \( A \)

\[ \mathcal{CN}(A\mu, A\Sigma A^H) \]

Reconstruction: \( A\Sigma A^H \) full covariance matrix for all pixels

\( \Rightarrow \) Not practical
Geometry Factor

Spatially dependent SNR:

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

Estimate using Monte-Carlo method by reconstructing Gaussian white noise \( n_j \).

\[
\sigma(x) = \sqrt{N^{-1} \sum_j |A^\dagger n_j|^2}
\]

(might be slightly biased)
Noise Amplification in Parallel Imaging

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

Parallel MRI as Inverse Problem

- Signal from multiple coils (image $x$, sensitivities $c_j$):

$$s_j(t) = \int_V d\vec{r} x(\vec{r}) c_j(\vec{r}) e^{-i\vec{r} \cdot \vec{k}(t)}$$

- Assumption: known sensitivities $c_j$
  ⇒ linear relation between image $x$ and data $y$

- Image reconstruction is a linear inverse problem:

$$Ax = y$$

Parallel MRI: Regularization

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

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

- Influence of the regularization parameter $\alpha$:
Parallel MRI: Nonlinear Regularization

- Good noise suppression
- Edge-preserving

⇒ Sparsity, nonlinear regularization

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

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

Project 1: Iterative SENSE

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

- **Tools**: Matlab, reconstruction toolbox, python, ...
- **Deadline**: Feb 24
- **Hand in**: Working code and plots/results with description.
- **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 \( \langle x, Ay \rangle = \langle A^Hx, y \rangle \) for random vectors \( x, y \)
Project 1: Iterative SENSE

Step 2: Implement Reconstruction

Landweber (gradient descent)\(^1\)

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

Conjugate gradient algorithm\(^2\)

Next lesson: Iterative Algorithms

Step 3: Experiments

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

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, IRGNM, ...)
- 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://www.eecs.berkeley.edu/~uecker/toolbox.html

- Requirements: FFTW, GSL, LAPACK (CUDA, ACML)

Ubuntu:
- `sudo apt-get install libfftw3-dev`
- `sudo apt-get install libgsl0-dev`
- `sudo apt-get install liblapack-dev`

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

Data files store multi-dimensional arrays.

example.hdr ← Text header
example.cfl ← 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
```

Load result into Matlab/Octave:

```
⟩ data = squeeze(readcfl('out'));
⟩ imshow3(abs(data), []);
```
C Programming Example

```c
#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