Computational MRI

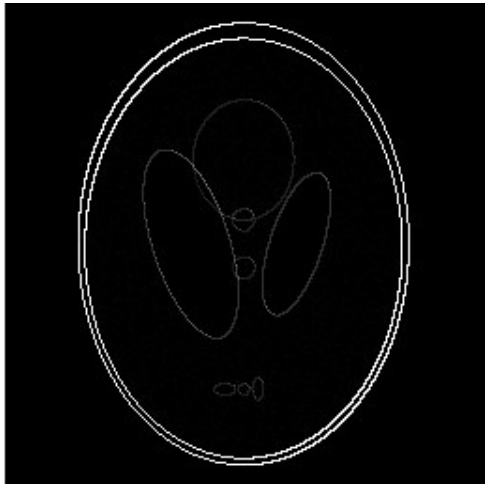
Compressed Sensing

Compressed sensing: The big picture

- Exploit image sparsity/compressibility to reconstruct undersampled data

Original (non-sparse)

Gradient (sparse)



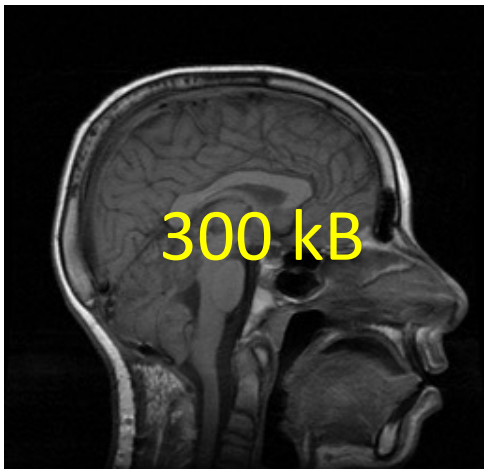
Which image would require fewer samples?

  - Nyquist: same FOV, same number of samples
  - Common sense: fewer non-zero pixels, fewer samples

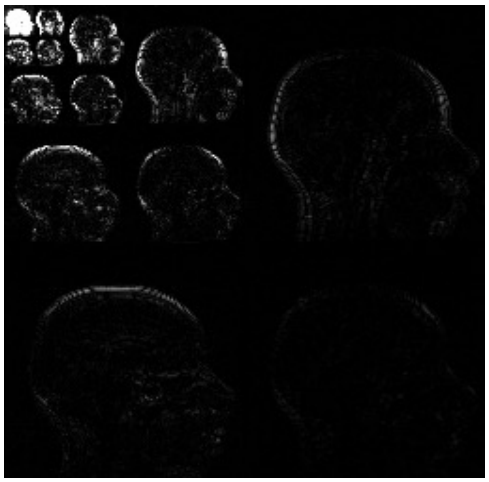
Image compression

- Essential tool for modern data storage and transmission
- Exploit pixel correlations to reduce number of bits
- First reconstruct, then compress

Fully-sampled acquisition
(Nyquist rate)

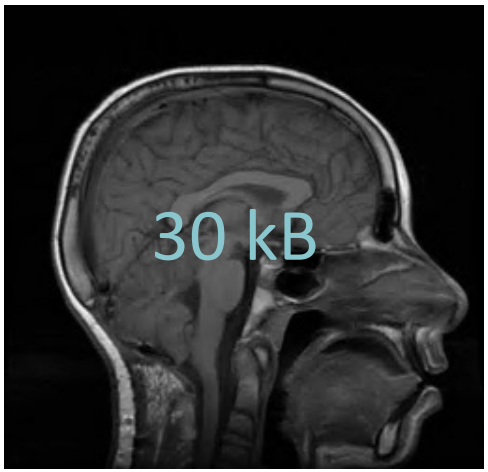


Sparsifying transform
(e.g. wavelets)



Store or transmit
non-zero coefficients only

Recover image from
sparse coefficients



10-fold compression

Nyquist sampling is inefficient

- Question
  - Why do we need to acquire samples at the Nyquist rate if we are going to throw away most of them?

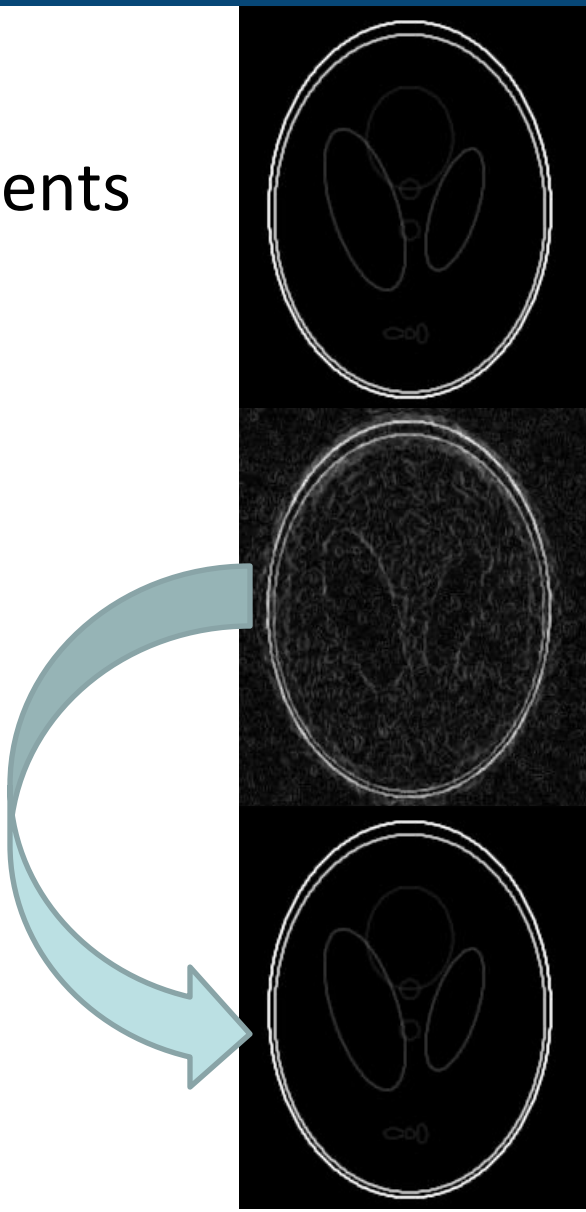
- Answer
  - We don't. Do **compressed sensing** instead
  - Build data compression in the acquisition
  - First compress, then reconstruct

Candès E,Romberg J,Tao T. IEEE Trans Inf Theory 2006; 52(2): 489-509

Donoho D. IEEE Trans Inf Theory 2006; 52(4): 1289-1306.

Compressed sensing components

- Sparsity
  - Represent images with a few coefficients
  - Transform: wavelets, gradient, etc.
- Incoherence
  - Noise-like aliasing artifacts
- Non-linear reconstruction
  - Remove aliasing artifacts



Simple compressed sensing example

k-space

No undersampling

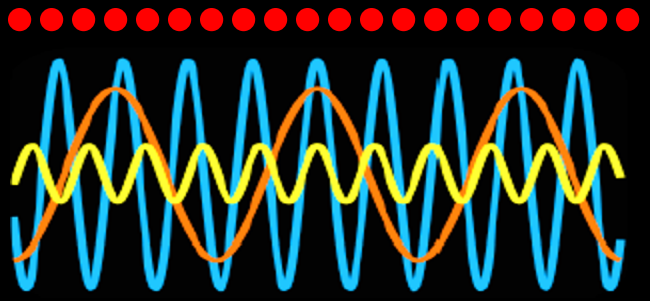
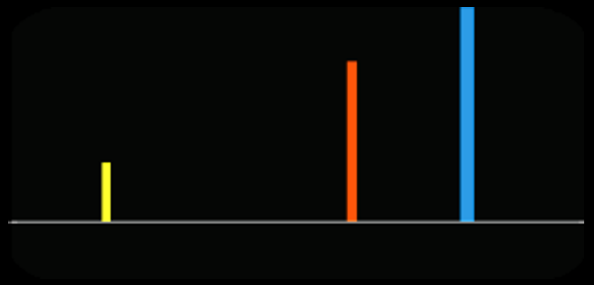
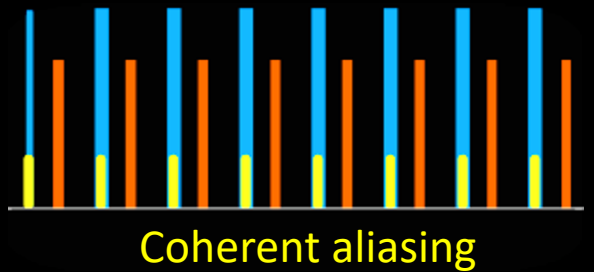


Image space

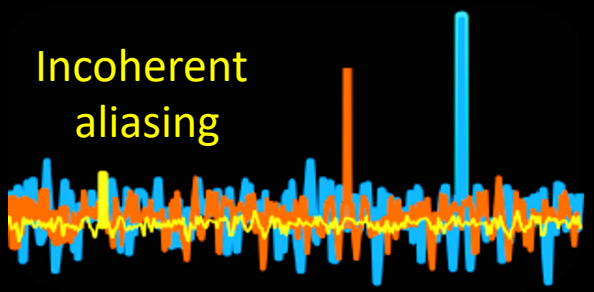


Regular undersampling



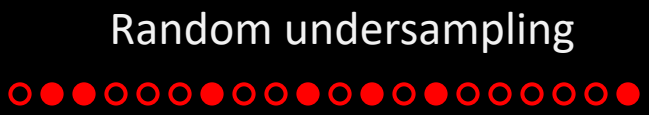
Coherent aliasing

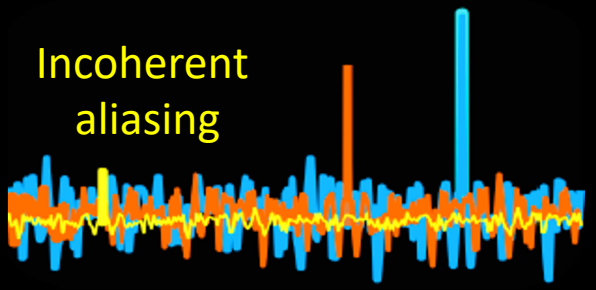
Random undersampling



Courtesy of Miki Lustig

Breakout session: Simple compressed sensing

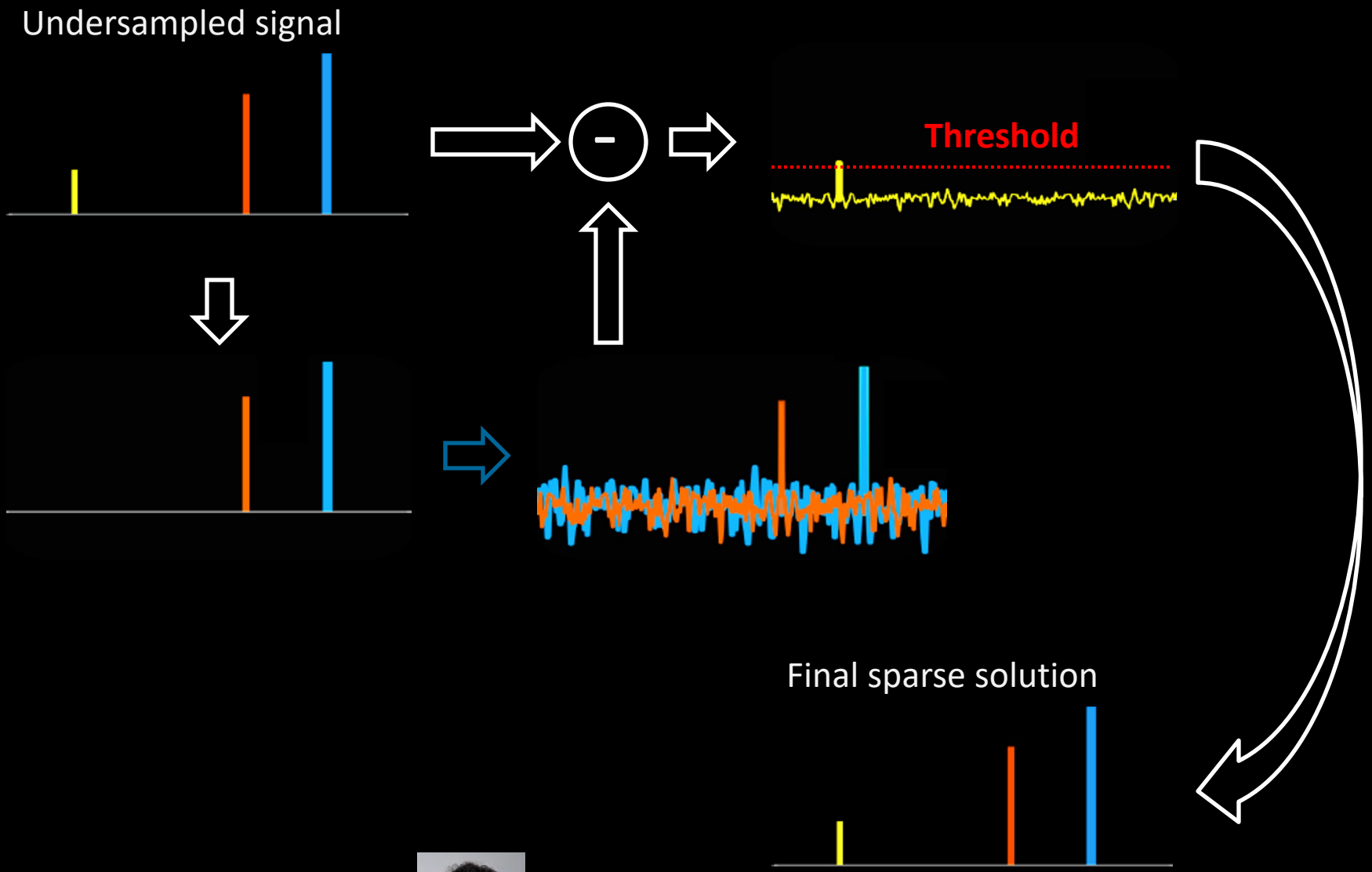




Find sparse solution: Simple iterative algorithm

- Threshold so that 2 largest impulses are preserved
- Apply undersampling
- Subtract from previous signal
- Threshold again

Simple compressed sensing example



Courtesy of Miki Lustig

Sparsity

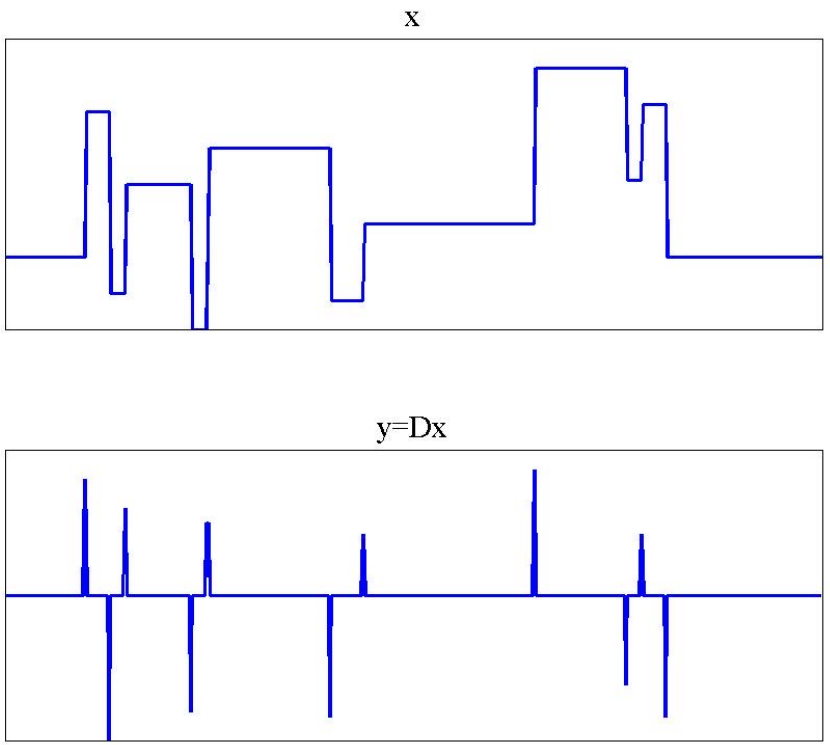
Sparsifying transforms

- Finite differences

$$y(n) = x(n) - x(n-1)$$

In matrix form : $y = Dx$

$$D = \begin{bmatrix} 1 & -1 & & \\ & 1 & -1 & \\ & & 1 & -1 \end{bmatrix}$$



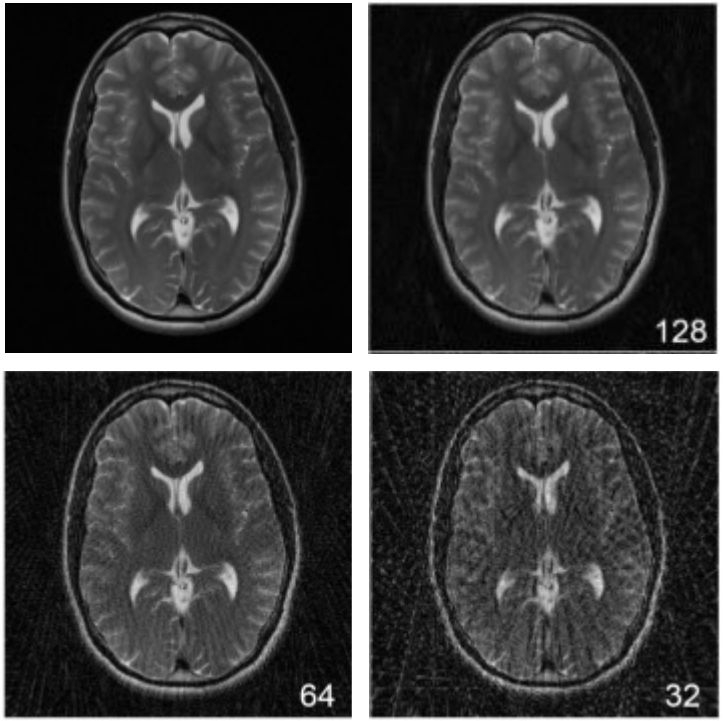
- Total variation

$$TV(x) = \sum_{n=2}^{N} |x(n) - x(n-1)|$$

$$\min TV(x) = \min \|Dx\|_1$$

Example: TV Norm, radial subsampling

Data set	Total Variation (a.u.)
Original (256×256)	2801

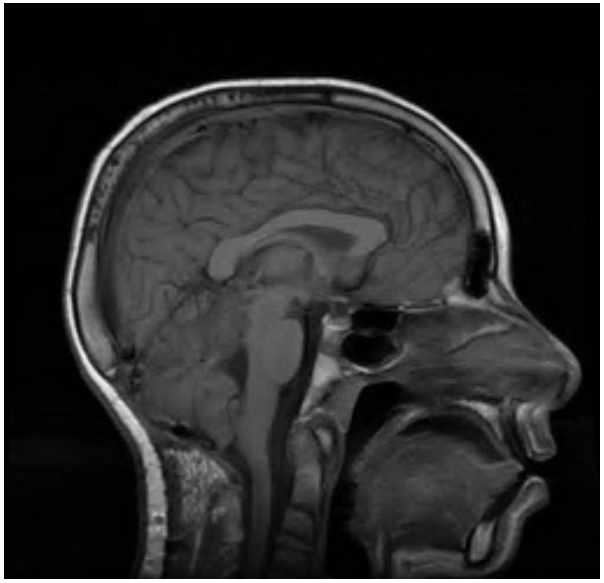


Rudin et al., Phys. D 60: 259-268 (1992)
Block et al., MRM 57: 1086-1098 (2007)
Knoll et al., MRM 65: 480-491 (2011)
Knoll et al., MRM 67: 43-41 (2012)

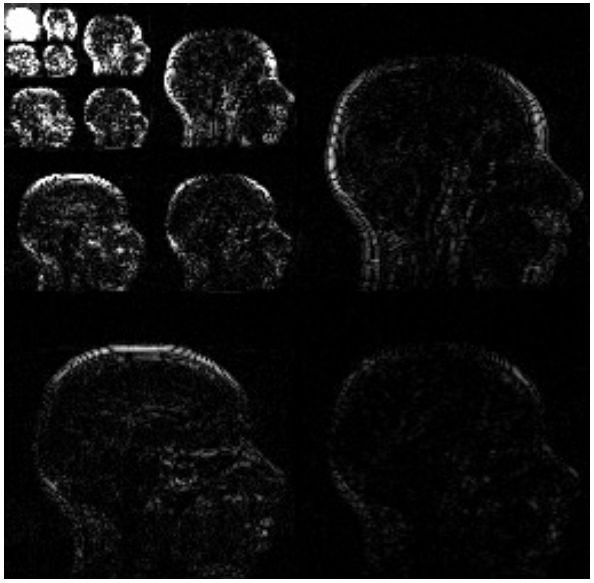
Sparsifying transforms

- Wavelets
  - Multiresolution image representation
  - Recursive application of the wavelet function modified by the scaling function (each resolution is twice of that of the previous scale)

Brain image



Daubechies 4-tap wavelet transform



Incoherence

Transform Point Spread Function (TPSF)

- Encoding model: $\mathbf{s} = \mathbf{E}\mathbf{m}$
- Representation model: $\mathbf{p} = \mathbf{W}\mathbf{m}$ (**p** is sparse)

W is orthogonal

$$\mathbf{s} = \mathbf{E}\mathbf{W}^H\mathbf{p}$$

- Transform point spread function for position *r*

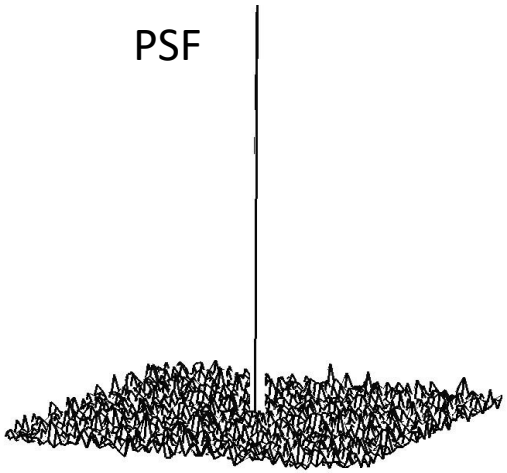
$$TPSF(r) = \mathbf{W}\mathbf{E}^H\mathbf{E}\mathbf{W}^H{}_{(r)}$$

Transform Point Spread Function (TPSF)

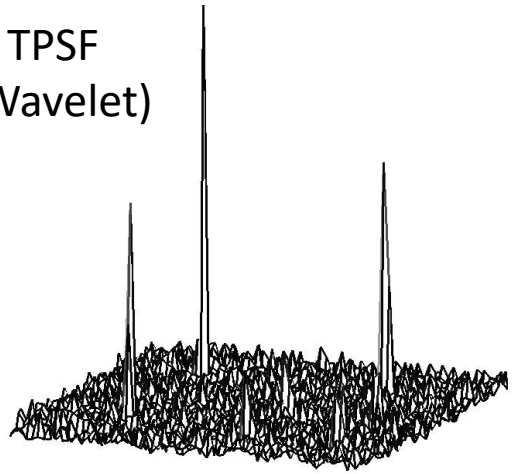
- Tool to check incoherence in the sparse domain
- Computation
  - Apply inverse FFT to sampling mask (1=sampled, 0=otherwise)
  - Apply sparsifying transform
- Incoherence = ratio of the main peak to the std of the pseudo-noise (incoherent artifacts)

Sampling mask

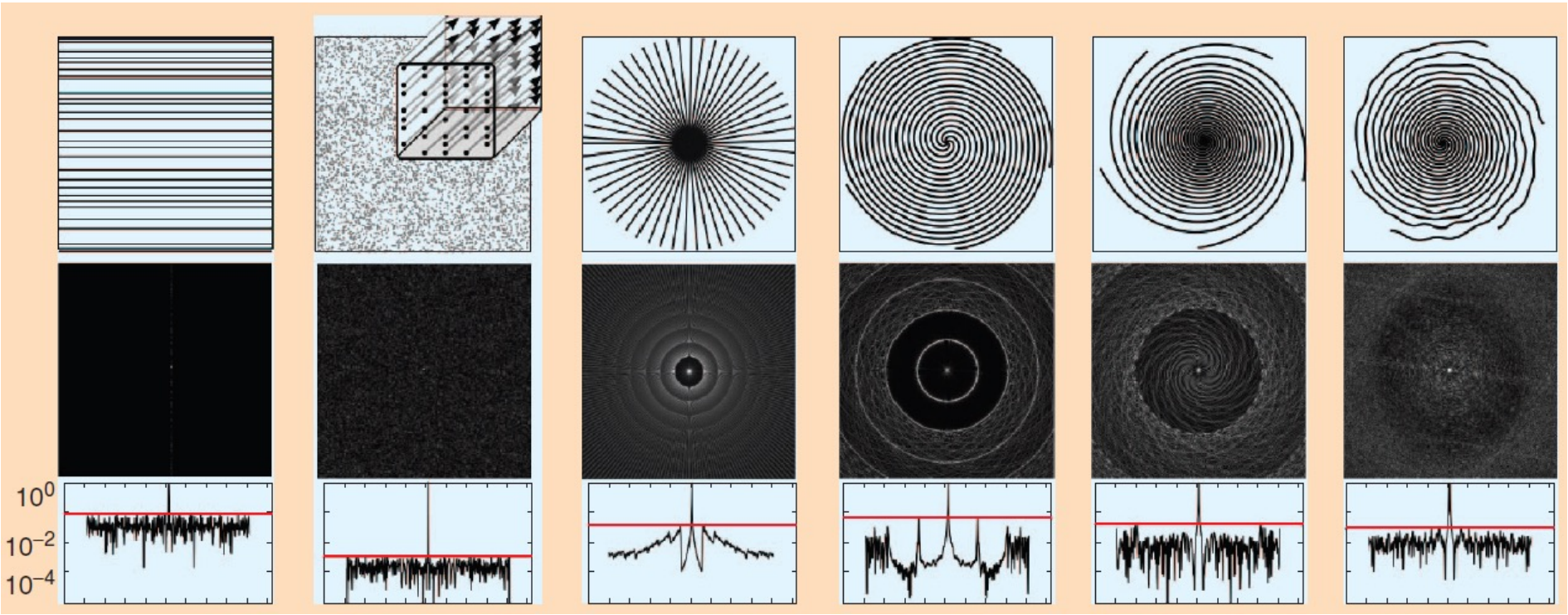
PSF



TPSF
(Wavelet)

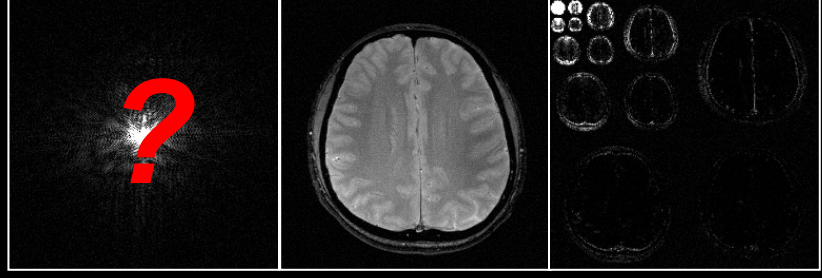


Incoherence of k-space trajectories

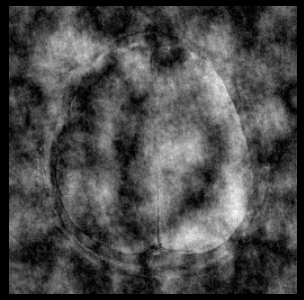
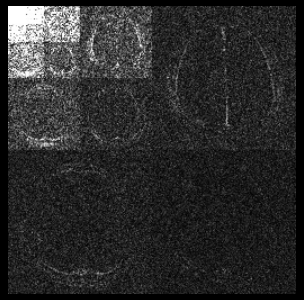
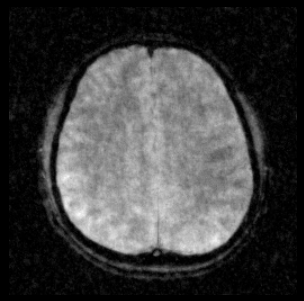
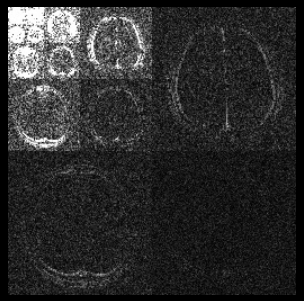
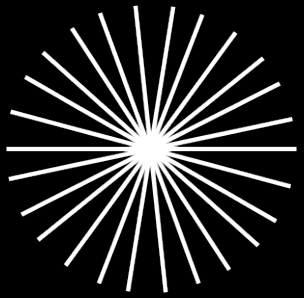
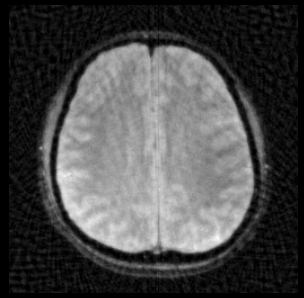
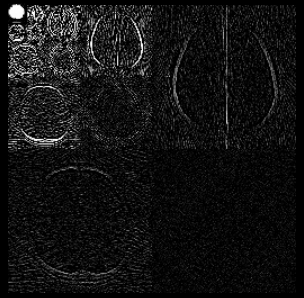


Lustig M, Donoho DL, Santos JM, Pauly JM. Compressed Sensing. MRI, IEEE Signal Processing Magazine, 2008; 25(2): 72-82

Incoherent sampling patterns



- What are the best samples?

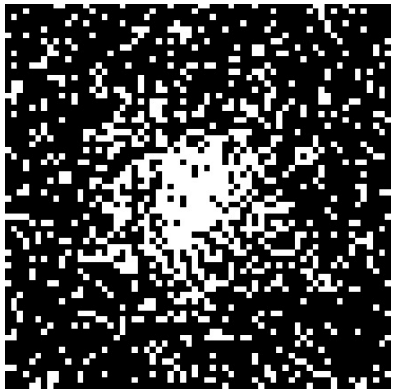
	Sampling pattern	Image domain	Sparse domain
Random undersampling (Cartesian)			
Variable-density random undersampling (Cartesian)			
Regular undersampling (radial)			

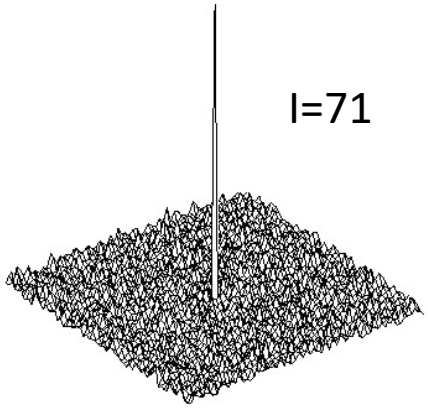
Dimensionality and incoherence

Sampling pattern
(R=4)

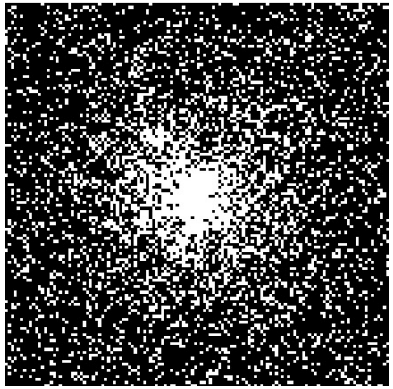
PSF

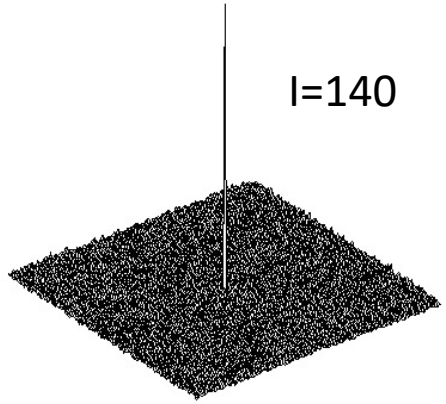
64x64
space





128x128
space





Sparse reconstruction

Compressed sensing reconstruction

- Acquisition model: $\mathbf{d} = \mathbf{E}\mathbf{m}$

d : acquired data
E : undersampled Fourier transform
m : image to reconstruct

- Sparsifying transform: $\mathbf{T}$

$$\min_{\mathbf{m}} \|\mathbf{T}\mathbf{m}\|_1 \text{ subject to } \|\mathbf{E}\mathbf{m} - \mathbf{d}\|_2 < \varepsilon$$

Sparsity

Data consistency

$$\|\mathbf{x}\|_1 = \sum_{n=1}^{N} |x_i| \quad (l_1\text{- norm of } \mathbf{x})$$

Compressed sensing reconstruction

- Unconstrained optimization (in practice)

$$\min_{\mathbf{m}} \left\| \mathbf{E}\mathbf{m} - \mathbf{d} \right\|_2^2 + \lambda \left\| \mathbf{T}\mathbf{m} \right\|_1$$

- Regularization parameter λ
  - Trade-off between data fidelity and removal of aliasing artifacts
  - High λ: artifact removal and denoising at the expense of image corruption (blurring, ringing, blocking, etc)
  - Low λ : no image corruption, but residual aliasing

Compressed sensing reconstruction

- Gradient descent
  - Cost function: $C(\mathbf{m}) = \left\|\mathbf{Em} - \mathbf{d}\right\|_2^2 + \lambda\left\|\mathbf{Tm}\right\|_1$
  - Iterations: $\mathbf{m}_{n+1} = \mathbf{m}_n - \alpha\nabla C(\mathbf{m}_n)$

$$\nabla C(\mathbf{m}_n) = 2\mathbf{E}^H\left(\mathbf{Em}_n - \mathbf{d}\right) + \lambda\mathbf{T}^H\mathbf{M}^{-1}\mathbf{Tm}_n$$

M is a diagonal matrix: $M_{ii} = \sqrt{(\mathbf{Tm})_i^*(\mathbf{Tm})_i + \mu}$

Approximate assumption: Regularized l1 norm

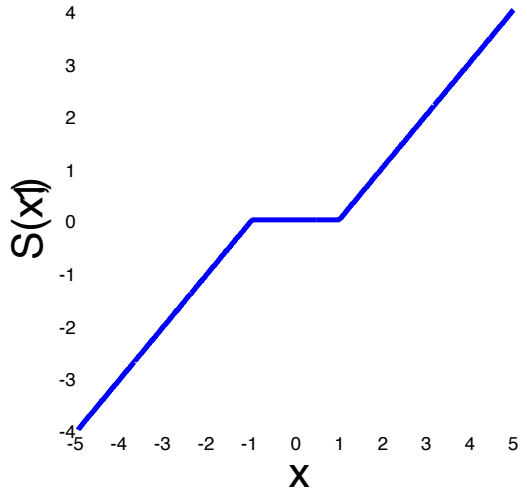
$$|x| = \sqrt{x^* x + \mu}$$

See lab exercise 7 for proof

Compressed sensing reconstruction

- Proximal gradient descent (iterative soft-thresholding)
  - Soft-thresholding operation

$$S(x,\lambda)=\begin{cases}0, \text{ if } |x|\le\lambda\\ \dfrac{x}{|x|}\left(|x|-\lambda\right), \text{ if } |x|>\lambda\end{cases}$$

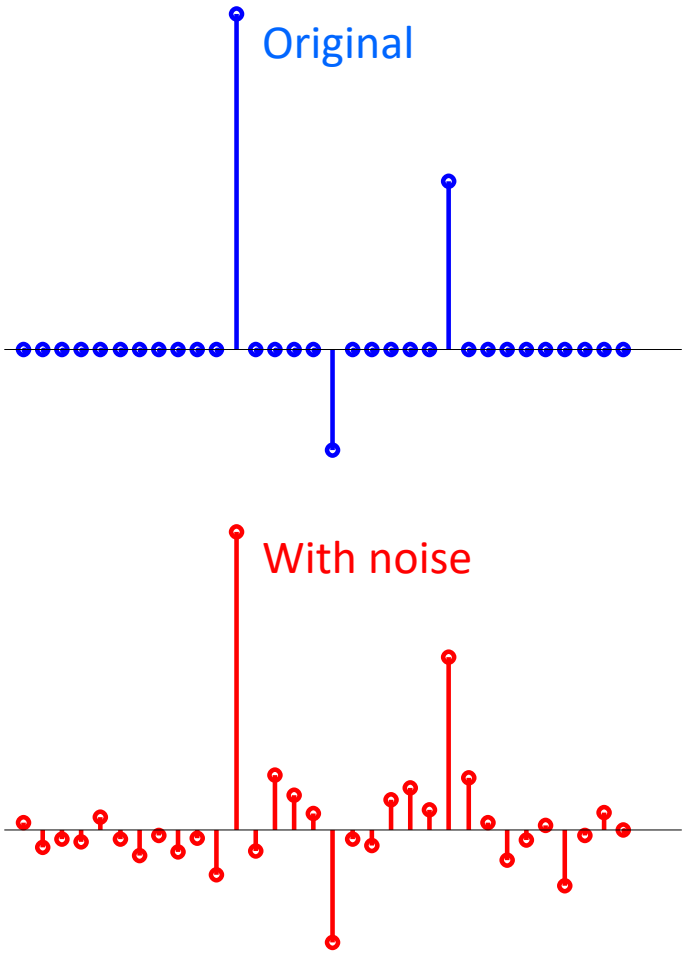


  - Our gradient gradient-descent algorithm becomes

$$\mathbf{m}_{n+1}=\mathbf{T}^{-1}\left[S\left(\mathbf{T}\left[\mathbf{m}_n-\mathbf{E}^H\left(\mathbf{E}\mathbf{m}_n-\mathbf{d}\right)\right],\lambda\right)\right]$$

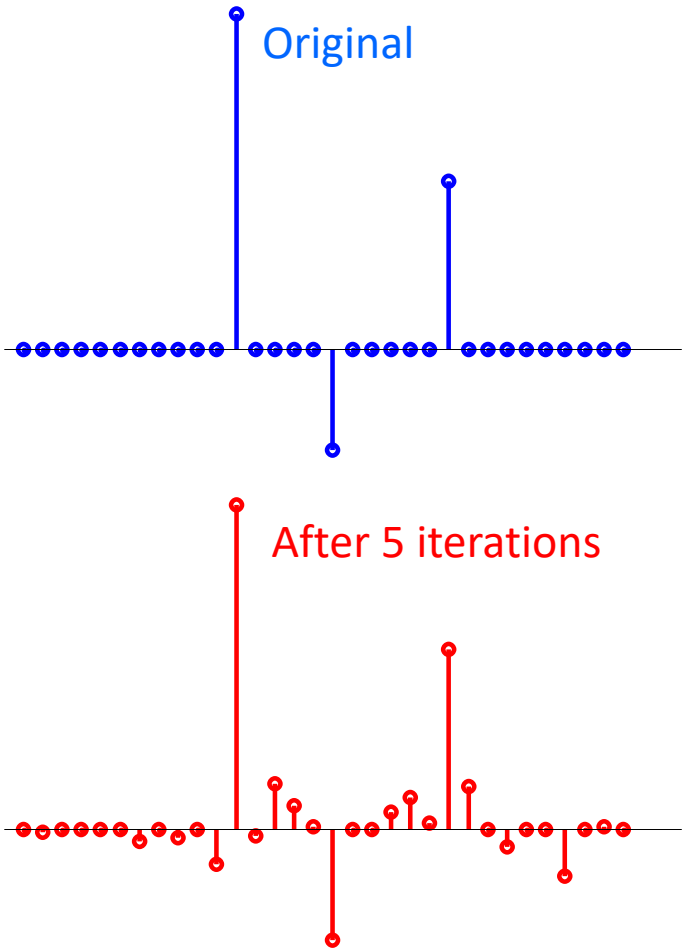
Iterative soft-thresholding

- Sparse signal denoising



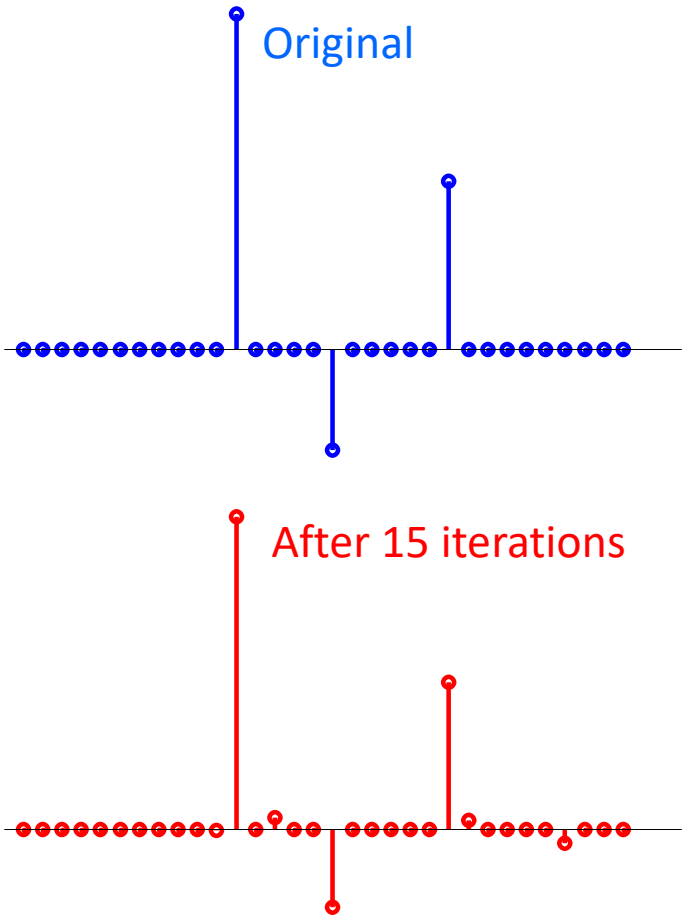
Iterative soft-thresholding

- Sparse signal denoising



Iterative soft-thresholding

- Sparse signal denoising



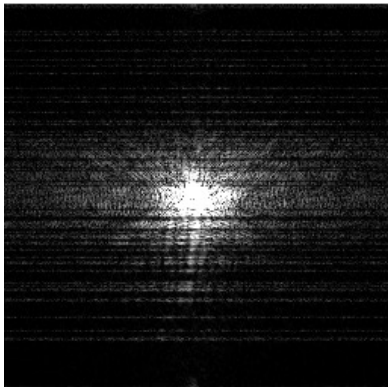
Iterative soft-thresholding in Matlab

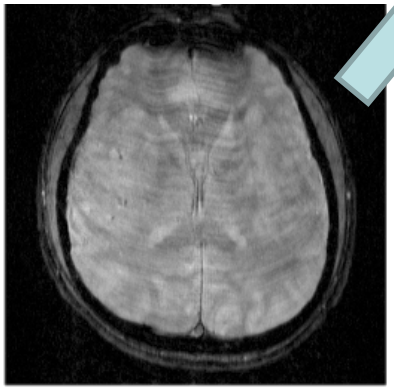
```
function m=cs_ista(d,lambda,nite)
% T and Ti are the forward and inverse sparsifying
% transforms
m = ifft2_mri(d); % initial solution
sm = s~=0; % sampling mask
for ite=1:nite,
      % enforce sparsity
      m = Ti (SoftT(T(m),lambda));
      % enforce data consistency
      m = m - ifft2_mri( (fft2_mri(m).*sm - d).*sm );
end

% soft-thresholding function
function y=SoftT(x,p)
      y = (abs(x)-p).*x./abs(x).*(abs(x)>p);
end
```

Iterative soft-thresholding

Initial solution

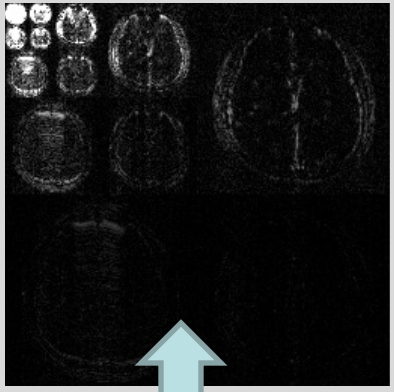
Inverse FT of the zero-filled k-space





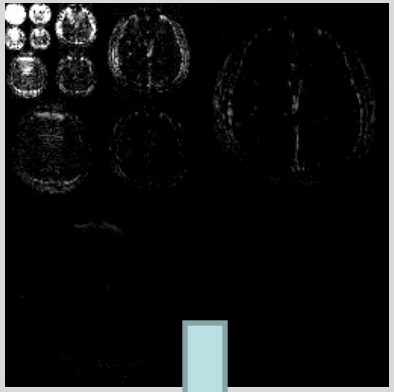
Iterations

Sparse representation

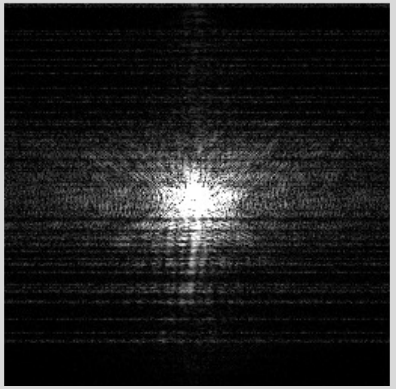


$S(w_i, \lambda)$

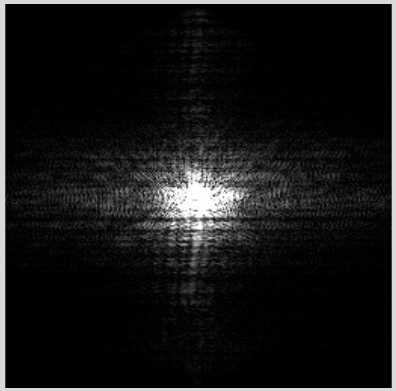
After soft-thresholding



k-space representation



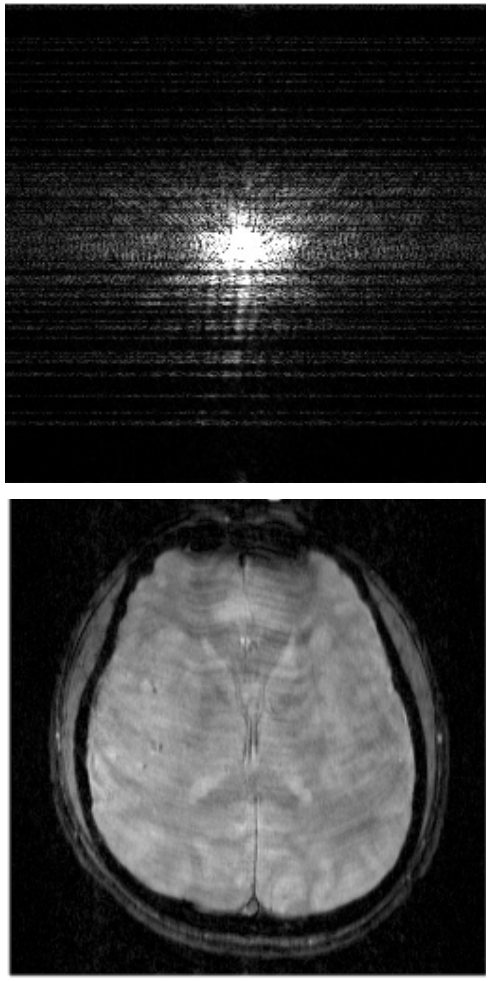
After data consistency



Iterative soft-thresholding (R=3)

Initial solution

Inverse FT of the zero-filled k-space



After 30 iterations

Iterative soft-thresholding

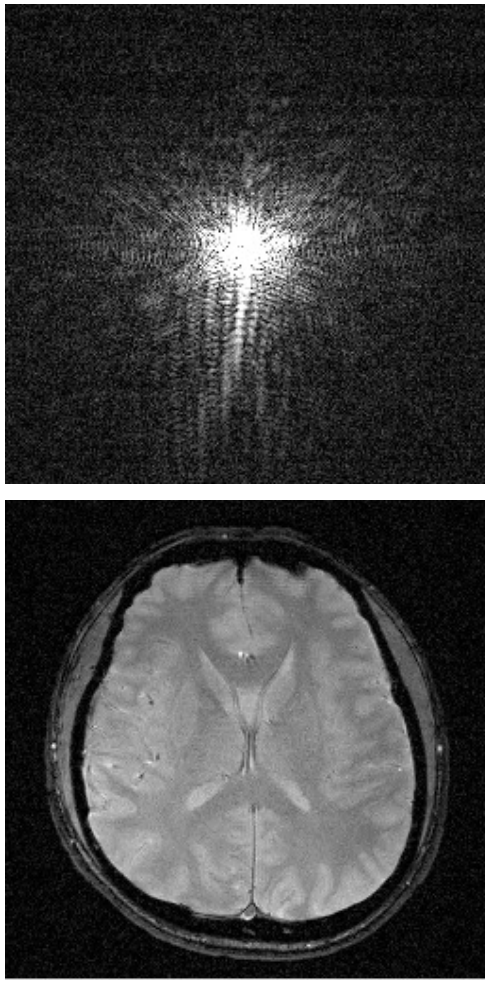
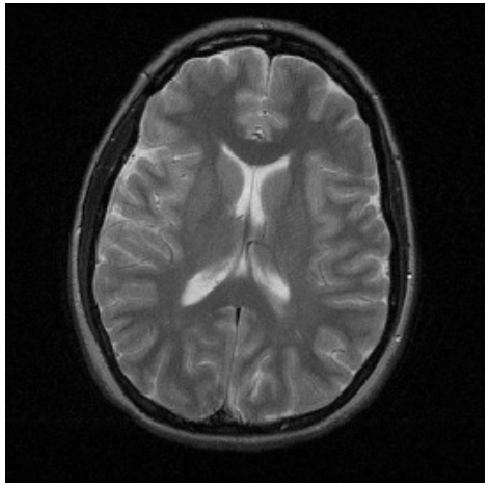


Image quality in compressed sensing

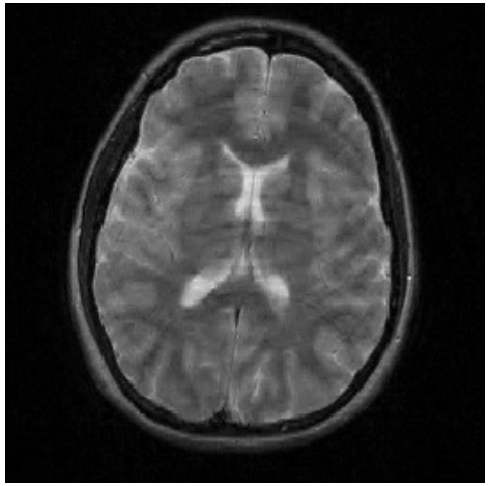
- SNR is not a good metric

- Loss of small coefficients in the sparse domain
  - Loss of contrast
  - Blurring
  - Blockiness
  - Ringing
  - Images look more synthetic

R=2



R=4



Combination of compressed sensing and parallel imaging

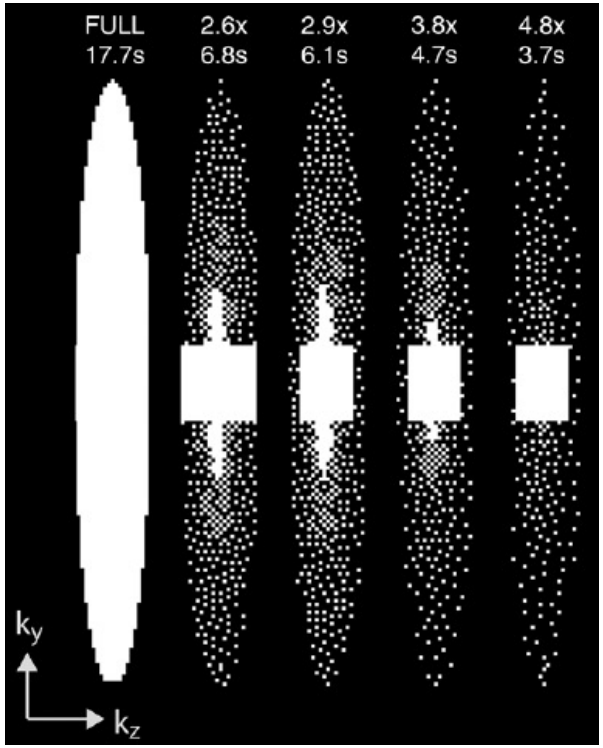
Why would CS & PI make sense?

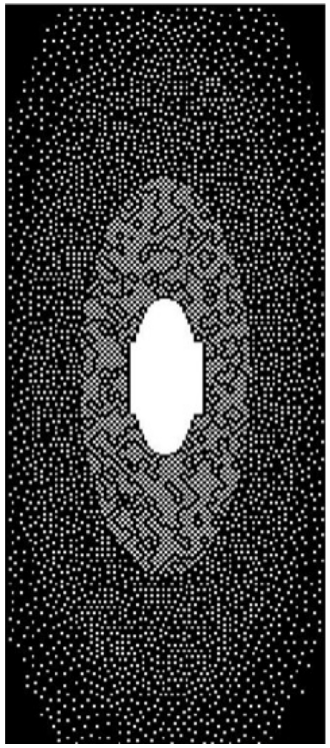
- Image sparsity and coil-sensitivity encoding are complementary sources of information
- Compressed sensing can regularize the inverse problem in parallel imaging
- Parallel imaging can reduce the incoherent aliasing artifacts

Challenges of CS & PI?

- CS requires irregular k-space sampling while PI requires regular k-space sampling

Poisson disk sampling





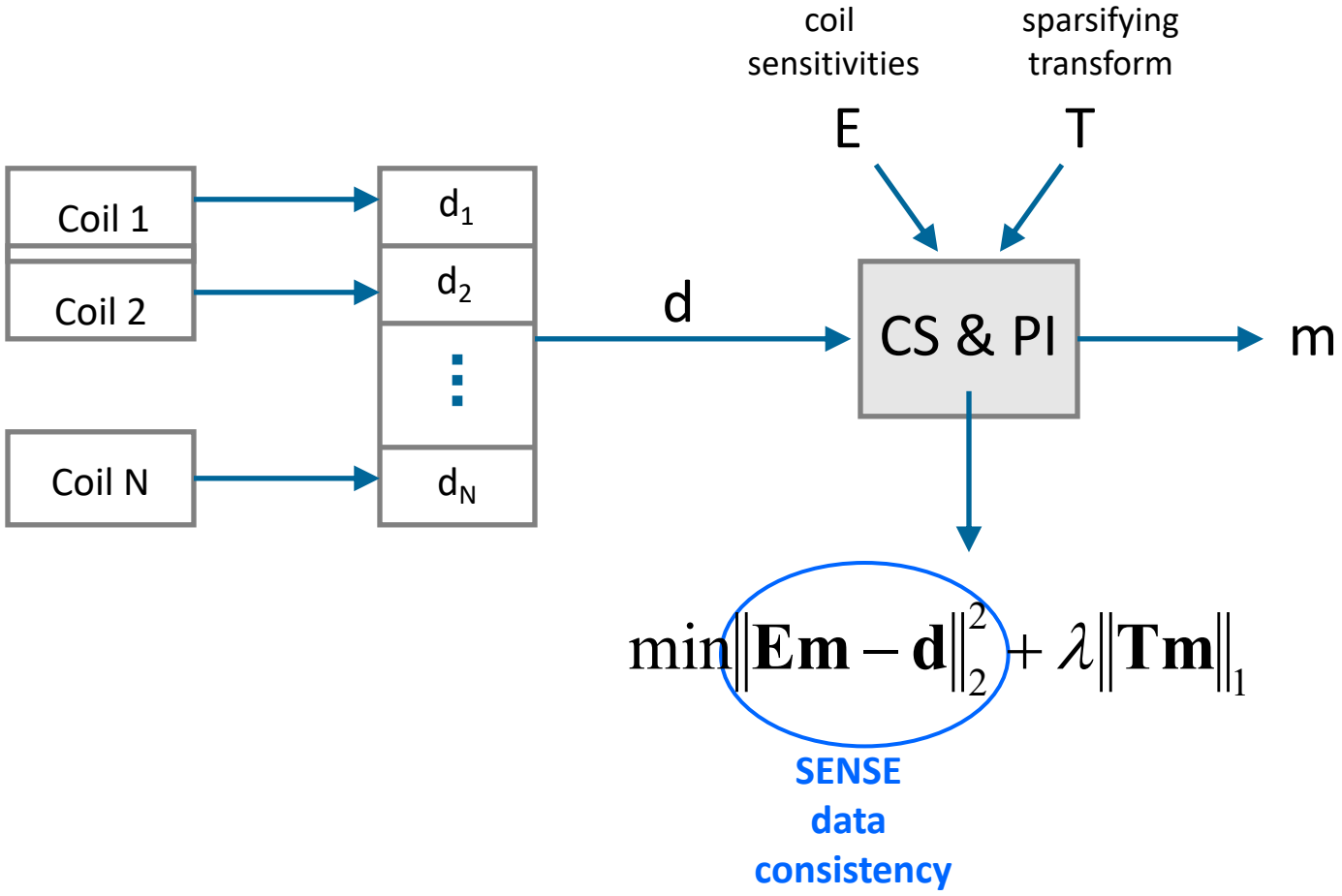
Approaches for CS&PI

- CS with SENSE parallel imaging model
  - Multicoil imaging with variational regularization
- CS with GRAPPA parallel imaging model
  - l_1-SPIRiT

Block et al. MRM 2007
Knoll et al. MRM 2011
Lustig M et al. MRM 2010

CS with SENSE parallel imaging model

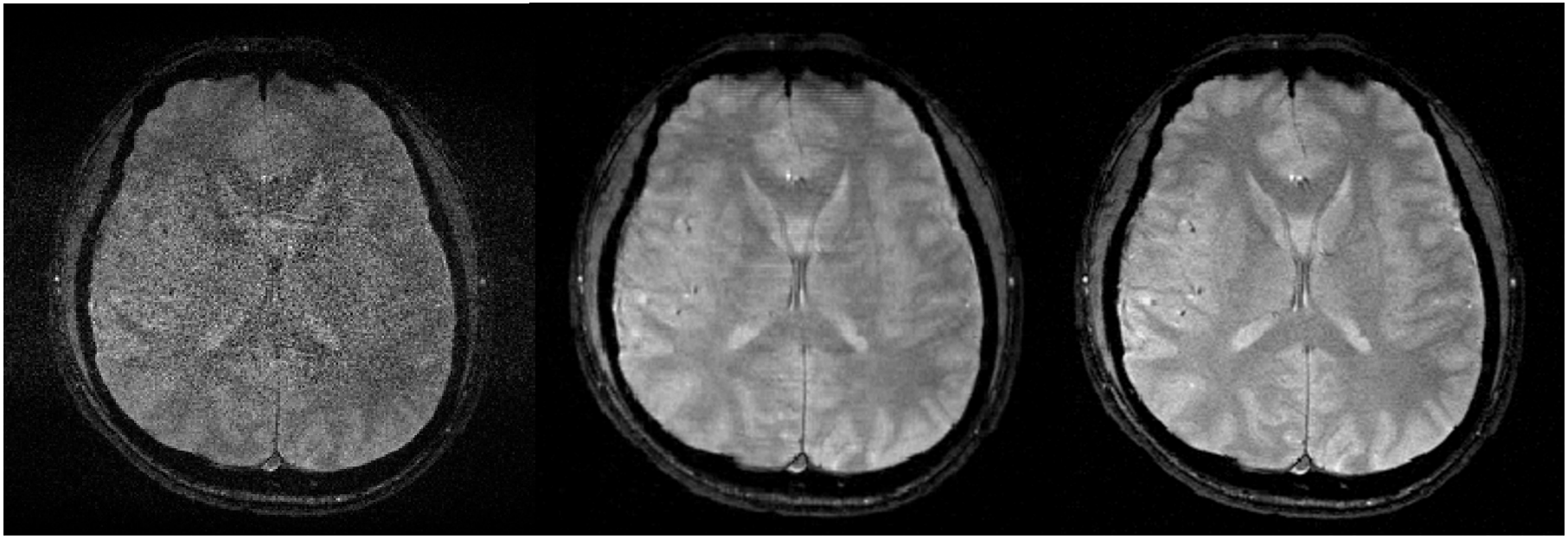
- Include parallel imaging in data consistency term



CS & PI for 2D imaging

- Siemens 3T Tim Trio
- 12-channel matrix coil array
- 4-fold acceleration

GRAPPA | Coil-by-coil CS | Joint CS & PI



Summary

- Compressed sensing
  - New sampling theorem
    - Information rate rather than pixel rate
  - Ingredients
    - Sparsity
    - Incoherence
    - Non-linear reconstruction

- Fast imaging tool for MRI
  - MR images are naturally compressible
  - Data acquisition in k-space facilitates incoherence
  - Can be combined with parallel imaging

Exercise

Computational MR imaging
Laboratory 8: Compressed Sensing

Report is due on Wednesday the week after the lab session at 23:59. Send your report by email to Bruno Riemenschneider (bruno.riemenschneider@fau.de) and Florian Knoll (florian.knoll@fau.de).

Learning objectives

- Refresh your linear algebra skills
- Apply compression transforms (e.g., wavelets) to obtain sparse representations of MR images
- Reconstruct randomly undersampled k-space data using compressed sensing approach

1. Sparsity/compressibility of brain images using the wavelet transform:

Medical images are generally not sparse, but they usually have a sparse representation after applying an appropriate transform. An example is the wavelet transform, which is the core transform used in the JPEG2000 standard. Wavelet coefficients are sub-band filters that hold both spatial (pixels) and frequency (k-space) information and thus they are able to represent an image with fewer non-zero coefficients.

- In this exercise we will use the Matlab wavelet toolbox. It has two functions: dwt2 and idwt2. These work much like fft2 and ifft2. The following shows example usage:

```
>> [cA,cH,cV,cD] = dwt2(X,'db4','mode','per');
>> X_recon = idwt2(cA, cH, cV, cD, 'db4', 'mode', 'per');
```

- The wavelet transform decomposes an image into low-pass (approximation coefficients) and high-pass (detail coefficients) features. cA has the approximation coefficients, which if you display them should look like a smoothed version of the original image. cH has detail coefficients in the horizontal direction, cV has detail coefficients in the vertical direction, and cD has detail coefficients in the diagonal direction. This particular code applies the Daubechies D4 wavelet transform (always a good one to start with – another good one is Haar) with periodic boundary conditions.

- Load the file data_lab6.mat and apply the wavelet transform to the fully-sampled data (kfull). The dwt2 and idwt2 functions only take real input, so you will have to split the image into its real and imaginary parts and then recombine them. Plot the magnitude of the brain image and its wavelet representation. Use a window from 0-1 for both images. Compute the l1-norm for both images. Which one is sparser?

- Compress the brain image by factors 5, 10 and 20 using the wavelet transform (hint: sort the wavelet coefficients in descending order using the sort function, compute the threshold T by finding the coefficient corresponding to n/compression-ratio and hard-threshold the wavelet transform using T). Plot the compressed image (scale: 0-1) and error image (scale: 0-0.1) and compute the RMSE for each compression ratios. Which compression ratio would you choose? What would be the maximum compressed sensing acceleration?

2. Compressed sensing reconstruction using iterative soft thresholding:

- Implement the iterative soft-thresholding approach discussed in class using the 4-tap Daubechies-type wavelet transform from exercise 1. Print the value of the cost function for each iteration in the command line.

```
cost = norm(fft2_mri(m).*sm - d,2)+lambda*norm(T(m),1);
fprintf('\n ite = %d, cost = %f',ite,cost);
```

- The variable kacc is an undersampled dataset using variable-density random undersampling. Plot the undersampling pattern and compute the acceleration factor.

- Reconstruct kacc using your iterative soft-thresholding algorithm. The algorithm requires choosing a value for lambda. A rule of thumb is to choose a threshold close to 1% of the maximum absolute value of the starting solution. Try your reconstructions with l=5%, 1% and 0.5%. Plot the initial solution, final reconstruction and corresponding error images with respect to the fully-sampled one.