SpRING: Sparse Reconstruction of Images using the Nullspace method and GRAPPA

D. S. Weller[1], J. R. Polimeni[2,3], L. Grady[4], L. L. Wald[2,3], E. Adalsteinsson[1], and V. Goyal[1]
[1]EECS, Massachusetts Institute of Technology, Cambridge, MA, United States, [2]A. A. Martinos Center, Dept. of Radiology, Massachusetts General Hospital, Charlestown, MA, United States, [3]Dept. of Radiology, Harvard Medical School, Boston, MA, United States, [4]Dept. of Image Analytics and Informatics, Siemens Corporate Research, Princeton, NJ, United States

Introduction

Accelerated parallel imaging methods like GRAPPA [1] leverage spatial weighting across coil sensitivities to undo coherent aliasing in undersampled data, at the expense of noise amplification. Compressed sensing (CS) [2-4] relies on incoherent sampling and nonlinear processing to reconstruct an image, at the expense of losing non-sparse details. Algorithms in [5-7] combine parallel imaging with CS to reconstruct high-SNR, approximately sparse images from undersampled data. The method presented in [5] is improved using the mixed ℓ_1/ℓ_2-norm simultaneous sparsity (over P coils) penalty function $\|w\|_S = \sum_n \|[w_{n,1}, ..., w_{n,P}]\|_2$ and the LSMR method [8] for solving least-squares problems, resulting in Sparse Reconstruction of Images using the Nullspace method and GRAPPA (SpRING). The *g*-factors factors, derived analytically for GRAPPA in [9], are computed empirically for GRAPPA, CS, and SpRING using the pseudo multiple replica method in [10].

Theory

Given undersampled *k*-space data d and GRAPPA reconstructed *k*-space $W(d)$, SpRING consists of finding the missing data x that minimizes $\|\text{diag}(C_1, ..., C_P) F^{-1}(K^T d+(K^c)^T x-W(d))\|^2 + \lambda\|\Psi F^{-1}(K^T d + (K^c)^T x)\|_S$. Here, F^{-1} is the inverse DFT, $[C_1, ..., C_P]$ are coil combination weights, K and K^c select the undersampled and missing *k*-space, respectively, and Ψ is the sparsifying transform. This method operates in the nullspace of the acquired data selection matrix K, guaranteeing the acquired data is preserved. The tuning parameter λ trades fidelity to the GRAPPA reconstruction for sparsity. The coil combination weights are computed from low-resolution estimates of the coil sensitivities $(S_1, ..., S_P)$ using the coil noise covariance matrix Λ: $[C_1(x,y,z), ..., C_P(x,y,z)] = \text{pinv}(\Lambda^{-1/2}[S_1(x,y,z), ..., S_P(x,y,z)]^T)\Lambda^{-1/2}$, where pinv() is the left pseudo-inverse.

Methods

The coil noise covariance matrix is measured from noise-only acquisitions, in which no excitation is applied, and the receivers sample thermal noise. The reference data is acquired with an un-accelerated T_1-weighted MPRAGE (256×256×176 sagittal slices, 1.0 mm isotropic resolution) using a Siemens 32-channel head coil on a Siemens Trio 3 T scanner. The data is split into axial slices, and each slice is cropped individually. The 250×170 cropped slice shown is uniformly undersampled by a factor of 4 in both directions, and a 36×36 block from the center is used as ACS lines for estimating the coil sensitivities and generating the GRAPPA kernel. The central ACS block is treated like known data during reconstruction. GRAPPA, nullspace CS (SpRING without the GRAPPA fidelity term), SpRING, and L_1 SPIR-iT [6] are used on the under-sampled data. The four-level 'db2' DWT sparsifying transform is used for CS, SpRING, and L_1 SPIR-iT. The reconstructed coil images are combined using the estimated combination weights. To compute the noise amplification of each algorithm, zero-mean white Gaussian noise, with covariance Λ across coils, is added to the *k*-space data, and the result is reconstructed using each algorithm. The *g*-factors are computed from these reconstructions using the method in [10], combined across coils with the estimated coil combination weights.

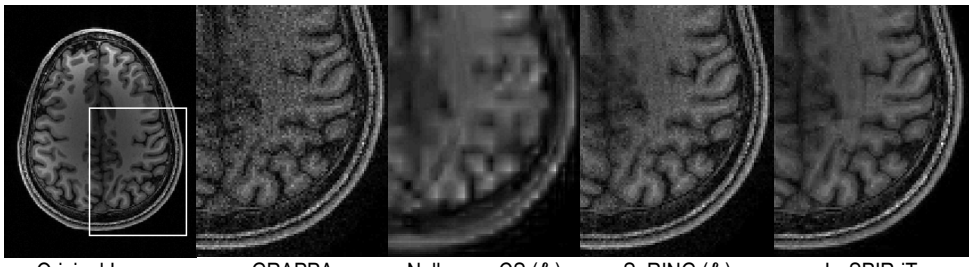

Fig. 1: The slice is uniformly undersampled by a factor of 4 in both directions and reconstructed using GRAPPA, nullspace CS, SpRING, and L_1 SPIR-iT. The ℓ_1/ℓ_2-norm penalty function is used for both nullspace CS and SpRING. The inset area is shown for each of the algorithms. The noise amplification in the GRAPPA result and the over-smoothing in the nullspace CS result are clearly visible. In SpRING and L_1 SPIR-iT, λ is chosen to minimize the RMSE of the magnitude image.

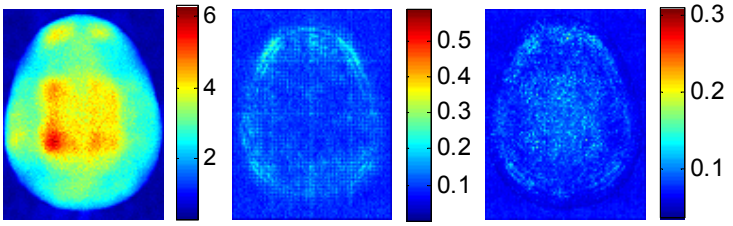

Fig. 2: The estimated *g*-factors are shown for the reconstruction algorithms using 350 Monte Carlo trials. The spatial average and maximum *g*-factors are shown for each reconstruction method. As expected, the GRAPPA noise amplification is greatest in the center of the image, and the nonlinear CS and SpRING algorithms' *g*-factors depend on the image intensities. (Note the different color scales used in each panel.)

Results

The SpRING and L_1 SPIR-iT reconstructions in Figure 1 clearly outperform either GRAPPA or CS alone. However, L_1 SPIR-iT does not mitigate the coherent aliasing as effectively in this example. The noise amplification *g*-factors for the GRAPPA, CS, and SpRING algorithms are plotted in Figure 2. The smoothing induced by the sparsity enforcement term results in *g*-factor values less than one. Whereas the largest *g*-factors for GRAPPA are concentrated near the center of the image (away from the coils), the *g*-factors for CS and SpRING are all greatest near the largest image intensities. Since the CS and SpRING algorithms are nonlinear, the *g*-factors are image-dependent and vary with the magnitude of the added noise. Both CS and SpRING do succeed in greatly suppressing the added noise, reducing both the mean and the maximum *g*-factors by more than a factor of 10 relative to GRAPPA.

Discussion

SpRING succeeds in leveraging the approximate sparsity of the image to de-noise the GRAPPA result and to recover the gray-white contrast lost in the CS-only result. By adjusting the tuning parameter, the sparsity of the result and the noise present can be controlled. SpRING also can be implemented for the GPU, where parallelization can greatly accelerate the Fourier and wavelet transforms in the implementation of the algorithm.

References

[1] Griswold et al., MRM 2002, 47(6): 1202-10. [2] Candès et al., IEEE Trans. IT 2006, 52(2): 489-509. [3] Donoho, IEEE Trans. IT 2006, 52(4): 1289-1306. [4] Lustig et al., MRM 2007, 58(6): 1182–1195. [5] Weller et al., ISMRM'10: 4880. [6] Murphy et al., ISMRM'10: 4854. [7] King et al., ISMRM'10: 4881. [8] Fong et al., Stanford: Report SOL 2010-2. [9] Breuer et al., MRM 2009, 62(3): 739-46. [10] Robson et al., MRM 2008, 60(4): 895-907.