USC–SIPI REPORT #458

PISCO Software Version 1.0

by

Rodrigo A. Lobos, Chin-Cheng Chan, Justin P. Haldar

March 2023

Signal and Image Processing Institute
UNIVERSITY OF SOUTHERN CALIFORNIA
USC Viterbi School of Engineering
Department of Electrical and Computer Engineering
3740 McClintock Avenue, Suite 400
Los Angeles, CA 90089-2564 U.S.A.

PISCO Software Version 1.0

Rodrigo A. Lobos, Chin-Cheng Chan, Justin P. Haldar

Signal and Image Processing Institute,
Ming Hsieh Department of Electrical and Computer Engineering,
University of Southern California

1 Introduction

The estimation of sensitivity maps from k-space calibration data is a common task in many multichannel MRI applications[1]. In the last decade, subspace-based estimation methods have gained popularity within the MRI community, where ESPIRiT [3] has emerged as the method of choice by many researchers. Even though these methods possess great estimation accuracy and robustness, they can be computationally demanding, and their underlying theoretical principles can be nontrivial to understand. In view of these limitations, we have proposed in [1,2] a novel theoretical framework for subspace-based sensitivity map estimation. This new framework relies on theoretical concepts from the literature on linear predictability and structured low-rank modeling, and we expect it might be more intuitive an easier to understand for some readers. Based on these novel theoretical concepts, we have proposed a nullspace-based algorithm for sensitivity map estimation which is theoretically equivalent to ESPIRiT. In addition, we have also introduced in [1,2] a set of computational techniques — which we collectively call PISCO (Power iteration over simultaneous patches, Interpolation, ellipSoidal kernels, and FFT-based COnvolution) — that, remarkably, can enable substantial improvements in computation time (up to a $100\times$-fold improvement in the cases we have tried) when integrated to subspace-based methods as shown in [1,2].

1.1 Software Availability

In order to provide the broader community with easy access to the theoretical and computational approaches in [1,2], we have released an open-source MATLAB software that implements the proposed nullspace-based algorithm for sensitivity map estimation using the PISCO computational techniques. The software is available at

http://mr.usc.edu/download/pisco/

The purpose of this report is to provide documentation for this software.

[1]We refer interested readers to [1,2] for a detailed mathematical description of the sensitivity map estimation problem. Comprehensive bibliography on sensitivity map estimation can be found in [1,2] as well.

2 Software Overview

The supplementary MATLAB code contains one main function called `PISCO_senseMaps_estimation.m` which implements the nullspace-based sensitivity map estimation algorithm presented in [1,2, Sect. III.D] using the PISCO computational techniques [1,2, Sect. IV]. The current implementation only covers sensitivity map estimation in the 2D case[2].

2.1 Quickstart

The simplest way to use the `PISCO_senseMaps_estimation.m` function is to call it with three arguments: a rectangular array of k-space calibration data `kCal`, and the desired size of the sensitivity maps `[N1, N2]`. This will run the nullspace-based algorithm using the PISCO techniques from [1,2] using default parameters, and is expected to work well in many cases of interest. However, the function also has a number of optional inputs that can be used to turn off some of the PISCO options (since, as described in [1,2], not every PISCO option is always computationally beneficial for every dataset) or to change PISCO parameter settings for cases where the defaults are suboptimal (note that the default values were designed based on the datasets considered in [1, 2] – for best performance, parameters should be adjusted for each new dataset).

2.2 Additional Inputs

In the following we explain the rest of the inputs of `PISCO_senseMaps_estimation.m` besides the aforementioned necessary inputs `kCal` and `[N1, N2]`.

- `tau`: Parameter (in Nyquist units) that determines the size of the k-space kernels used in the construction of the matrix **C**. For a rectangular kernel the associated neighborhood has size equal to `(2*tau+1) * (2*tau+1)`. For an ellipsoidal kernel the associated neighborhood has radius equal to `tau`, and can be seen as a rectangular neighborhood which corners were removed. In both cases increasing the value of `tau` can improve estimation quality, however, computational complexity would also increase. In addition, large values of `tau` can be detrimental in the presence of noise, and moderate sizes are recommended in order to avoid overfitting [4]. The default value corresponds to `tau = 3` if not specified, which for the examples shown in [1,2] offered a good trade-off between estimation quality and computational complexity.

- `threshold`: Specifies how small a singular value needs to be (relative to the maximum singular value) before its associated singular vector is considered to be in the nullspace of the matrix **C**. If this value is chosen to be too small or too big, then the nullspace of the matrix **C** might not be estimated correctly, which would negatively affect sensitivity map estimation. Heuristically, this value can be chosen as the point where the singular-values curve of the matrix **C** starts to flatten out. The default value corresponds to 0.05 if not specified.

- `kernel_shape`: Binary variable which indicates the shape adopted for the kernels that is used in the sensitivity map estimation process. This variable can be interpreted as a flag that "turns-on" the PISCO technique indicated in [1, 2, Sect. IV.B]. If equal to 0, then a rectangular shape is adopted for the kernels. If equal to 1, then an ellipsoidal shape is

[2]Extensions to the 3D case will be considered in future releases of this software.

adopted for the kernels. As indicated in [1, 2, Sect. IV.B], using ellipsoidal kernels allows substantial improvements in computational complexity with negligible effects on sensitivity estimation quality in comparison to using rectangular kernels. The default value of this variable corresponds to 1 if not specified.

- `FFT_nullspace_C_calculation`: Binary variable that indicates how the matrix $\mathbf{C}^H\mathbf{C}$ is calculated before calculating its nullspace (which corresponds to step (2) in the nullspace based algorithm proposed in [1, 2, Sect. III.D]). This variable can be interpreted as a flag that "turns-on" the PISCO technique indicated in [1, 2, Sect. IV.C]. If equal to 0, then $\mathbf{C}^H\mathbf{C}$ is calculated by calculating $\mathbf{C}$ first. If equal to 1, then $\mathbf{C}^H\mathbf{C}$ is calculated directly using FFT-based convolutions. It should be noted that turning on the PISCO technique associated to this variable can improve computation time and memory usage substantially, however, there are cases where this technique might not be beneficial as indicated in [1, 2].

- `PowerIteration_G_nullspace_vectors`: Binary variable that indicates how the last nullspace vector (i.e., the one associated to the smallest eigenvalue) of each matrix $\mathbf{G}(\mathbf{x})$ is calculated for each spatial location $\mathbf{x}$. This variable can be interpreted as a flag that "turns-on" the PISCO technique indicated in [1,2, Sect. IV.E]. If equal to 0, then the last nullspace vector of each matrix $\mathbf{G}(\mathbf{x})$ is calculated individually using SVD. If equal to 1, the last nullspace vectors of all the $\mathbf{G}(\mathbf{x})$ matrices are calculated jointly using a PowerIteration-based method. Turning on this PISCO technique usually allows substantial improvements in computation time in comparison to the SVD-based conventional approach. However, it should be noted that the fast convergence of power iteration is observed in cases where there is a big gap between the last two eigenvalues of $\mathbf{G}(\mathbf{x})$ [5], as in the cases studied in [1, 2]. In these cases a small number of iterations is needed, as shown in [1, 2], where only 10 iterations were considered. Nevertheless, there might be cases where the last two eigenvalues of $\mathbf{G}(\mathbf{x})$ are similar, which would negatively affect the convergence speed of power iteration. These cases are typically observed at locations where there exists aliasing, which correspond to cases where more than one sensitivity map is needed for an accurate description [1–3]. In these cases we could "turn-off" the PowerIteration-based technique by setting `PowerIteration_G_nullspace_vectors = 0`. The default value of this variable corresponds to 1 if not specified.

- `M`: Corresponds to the number of iterations used if the PowerIteration-based method is used to calculate the nullspace vectors of the $\mathbf{G}(\mathbf{x})$ matrices (i.e., when `PowerIteration_G_nullspace_vectors = 1`). As indicated in the previous point, a small number of iteration is needed for power iteration in cases where there is a big gap between the last two eigenvalues of $\mathbf{G}(\mathbf{x})$, and a big number of iteration is needed in cases where the last two eigenvalues are close to each other (e.g., when there exists aliasing in some spatial locations). The default value of this variable corresponds to 20 if not specified.

- `FFT_interpolation`: Binary variable that indicates if FFT-based periodic sinc interpolation is used in the sensitivity map estimation process. This variable can be interpreted as a flag that "turns-on" the PISCO technique indicated in [1, 2, Sect. IV.D]. Using this technique sensitivity maps are first estimated in a low-resoluton grid and then interpolated to a grid with nominal spatial resolution. This allows massive improvements in computation time and memory usage. In order for the FFT-based periodic sinc interpolation technique to be successful, there is the underlying assumption that sensitivity maps should be smooth. Given that sensitivity maps are complex, smoothness properties require sensitivity maps to have smooth magnitude and smooth phase (as in the examples studied in [1, 2]). Even though smooth magnitude is usually encountered, non-smooth phase patterns might be observed in sensitivity maps. For example, non-smooth phase patterns might be observed in cases where the original coil-array information has been preprocessed using coil-compression techniques [6]. This might affect the performance of the FFT-based periodic sinc interpolation technique, producing undesired discontinuities in the estimated sensitivity maps. In this case we can "turn-off" the FFT-based periodic sinc interpolation technique by setting `FFT_interpolation = 0`. The default value of this variable corresponds to 1 if not specified.

- `interp_zp`: Amount of zero-padding employed to create the low-resolution grid needed when the FFT-based periodic sinc interpolation technique is used in the sensitivity map estimation process (i.e., when `FFT_interpolation = 1`). If `[N1_cal, N2_cal]` correspond to the dimensions of the calibration data, then the low-resolution grid has dimensions `[N1_cal + interp_zp, N2_cal + interp_zp]`. Increasing the zero-padding improves sensitivity map estimation quality at the expense of increasing the computation time. The default value for this variable corresponds to 24 if not specified, which for the examples shown in [1, 2] offered a good trade-off between estimation quality and computational complexity.

- `gauss_win_param`: Parameter related to the Gaussian apodizing window that is used to obtain the smooth-phase low-resolution reconstruction needed in the FFT-based periodic sinc interpolation technique. This variable corresponds to the reciprocal value of the standard deviation of the Gaussian window. An incorrect choice of this parameter might produce a low-resolution reconstruction with nonsmooth-phase characteristics, which subsequently might cause the presence of undesired discontinuities in the estimated sensitivity maps. The default value for this variable corresponds to 100, which for the examples shown in [1, 2] allowed obtaining high-quality sensitivity maps when using the FFT-based periodic sinc interpolation technique.

2.3 Outputs

In the following we describe the outputs of `PISCO_senseMaps_estimation.m`.

- `senseMaps`: The estimated sensitivity maps. This variable corresponds to an array with dimensions $N_1 \times N_2 \times N_c$, where N_1, N_2 determine the size of the sensitivity maps, and N_c is the number of channels.

- `eigenValues`: Spatial maps for the eigenvalues of the $\mathbf{G}(\mathbf{x})$ matrices[3]. This variable corresponds to an array with dimensions $N_1 \times N_2 \times N_c$, where N_1, N_2 determine the number of spatial locations, and N_c is the number of eigenvalues (which is equal to the number of channels) of each matrix. Specifically, the entry `eigenValues(i,j,k)` would correspond to the k-th eigenvalue of the $\mathbf{G}(\mathbf{x})$ matrix where $\mathbf{x} = (i, j)$.

  If `PowerIteration_G_nullspace_vectors = 1`, only the last eigenvalue of the matrices $\mathbf{G}(\mathbf{x})$ is returned. In this case the dimensions of `eigenValues` are $N_1 \times N_2 \times 1$.

[3]These matrices have been normalized by the kernel size in order to have eigenvalues between 0 and 1 as indicated in [1, 2].

If `FFT_interpolation = 1`, then approximations of the eigenvalues of the matrices $\mathbf{G}(\mathbf{x})$ are returned.

As explained in the following section, `eigenValues` can be used to construct a mask for the support of the image in case the user wants to mask the estimated sensitivity maps. This is an optional step in the nullspace-based sensitivity map estimation algorithm as indicated in [1, 2, Sect. III.D].

3 Examples and Usage Recommendations

PISCO provides different options for sensitivity map estimation, that depend on which computational techniques are selected by the user. In the following we show two cases as examples: selecting all the PISCO computational techniques, and selecting none of the PISCO computational techniques[4]. These examples show that using the PISCO techniques allows massive improvements in computation time without sacrificing estimation quality. In the following experiments we used the $256 \times 256 \times 32$ Brain MPRAGE dataset described in [1,2, Sect. III.C] (images for 16 of the 32 channels are shown in Fig. 1), and we estimated sensitivity maps using the nullspace-based algorithm presented in [1,2, Sect. III.D]. In both cases sensitivity maps were estimated from Nyquist-sampled calibration k-space data of size 32×32 located in the center; a 7×7 FIR filter support Λ (which corresponds to selecting `tau = 3` as one of the inputs of `PISCO_senseMaps_estimation.m`); and a singular value threshold equal to 0.08 when calculating the nullspace of the matrix $\mathbf{C}$. We performed all computations on a MacBook Pro laptop computer with a 3.1 GHz Intel Core i7 processor and 16GB RAM. All of the following results can be easily replicated by running the script `example.m` provided with the software package.

In the second and third rows of Fig. 1 we show estimated sensitivity maps for 16 representative coils when all the PISCO techniques are used (Nullspace + PISCO), and when none of the PISCO techniques are used (Nullspace - PISCO). We can observe that both methods produced similar sensitivity maps, with no noticeable visual differences within the support of the image. It should be remembered that sensitivity maps are unidentifiable outside the support of the image [1,2, Sect. II], therefore, differences between the two configurations outside the support of the image should not be relevant. Remarkably, using all the PISCO techniques allowed estimating sensitivity maps in ~ 2.3 secs, which is considerably faster than the ~ 34.8 secs needed when none of the PISCO techniques were used.

For better visualization we also provide a masked version of these sets of sensitivity maps using a mask for the support of the image. We can see in the fourth and fifth rows of Fig. 1 that both methods obtained sensitivity maps with no noticeable differences after masking. It should be highlighted that masking of the sensitivity maps has been indicated as an optional step in the nullspace-based algorithm proposed in [1,2, Sect. III.D, step (6)]. This step is optional because the masked sensitivity maps are equivalent to the unmasked sensitivity maps from a data-consistency perspective. We have masked the sensitivity maps analogously to what is done in [3]. Specifically, we have zeroed-out entries where the last eigenvalue of the $\mathbf{G}(\mathbf{x})$ matrix is higher than a threshold close to zero (0.05 was selected for the cases shown in Fig. 1).

[4]In this case all the PISCO techniques are deactivated except by the PISCO technique which calculates the $\mathbf{G}(\mathbf{x})$ matrices directly [1, 2, Sect. IV.C], which is always activated in our implementation.

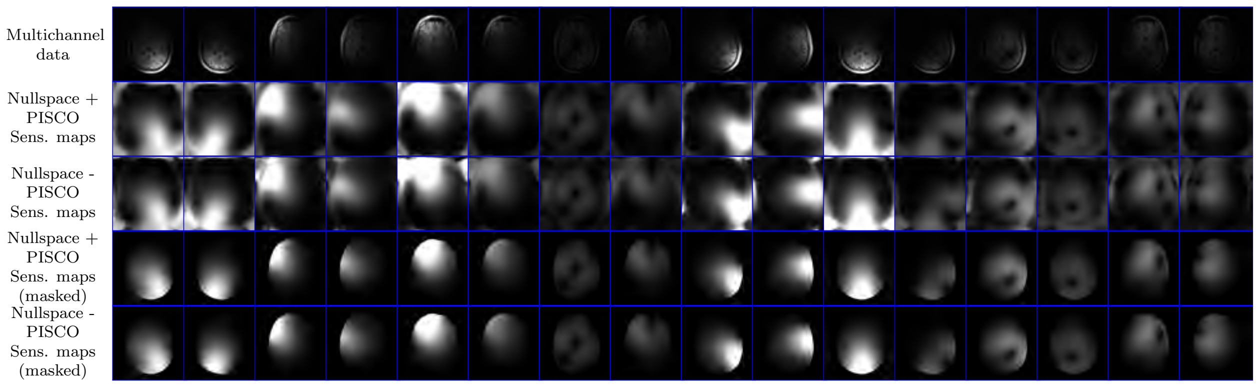
Multichannel data
Nullspace + PISCO Sens. maps
Nullspace - PISCO Sens. maps
Nullspace + PISCO Sens. maps (masked)
Nullspace - PISCO Sens. maps (masked)

Figure 1: Multichannel data used in our experiments (first row) and sensitivity maps estimated using the nullspace-based algorithm with and without the PISCO computational techniques. Estimated sensitivity maps are shown before and after applying a mask based on the support of the underlying image.

References

[1] R. A. Lobos, C.-C. Chan, and J. P. Haldar, "New theory and faster computations for subspace-based sensitivity map estimation in multichannel MRI," Submitted.

[2] R. A. Lobos, C.-C. Chan, and J. P. Haldar, "Extended version of 'New theory and faster computations for subspace-based sensitivity map estimation in multichannel MRI'," 2023, arXiv:2302.13431.

[3] M. Uecker, P. Lai, M. J. Murphy, P. Virtue, M. Elad, J. M. Pauly, S. S. Vasanawala, and M. Lustig, "ESPIRiT – an eigenvalue approach to autocalibrating parallel MRI: Where SENSE meets GRAPPA," *Magn. Reson. Med.*, vol. 71, pp. 990–1001, 2014.

[4] R. A. Lobos and J. P. Haldar, "On the shape of convolution kernels in MRI reconstruction: Rectangles versus ellipsoids," *Magn. Reson. Med.*, vol. 87, pp. 2989–2996, 2022.

[5] G. Golub and C. van Loan, *Matrix Computations*, 3rd ed. London: The Johns Hopkins University Press, 1996.

[6] D. Kim, S. F. Cauley, K. S. Nayak, R. M. Leahy, and J. P. Haldar, "Region-optimized virtual (ROVir) coils: Localization and/or suppression of spatial regions using sensor-domain beamforming," *Magn. Reson. Med.*, vol. 86, pp. 197–212, 2021.