Received: 2 July 2023 | Revised: 11 November 2023 | Accepted: 13 November 2023

DOI: 10.1002/mrm.29947

RESEARCH ARTICLE

Magnetic Resonance in Medicine

Replication of the bSTAR sequence and open-source implementation

Nam G. Lee[1,2] | **Grzegorz Bauman**[3,4] | **Oliver Bieri**[3,4] | **Krishna S. Nayak**[1,2]

[1]Alfred E. Mann Department of Biomedical Engineering, University of Southern California, Los Angeles, California, USA

[2]Ming Hsieh Department of Electrical and Computer Engineering, University of Southern California, Los Angeles, California, USA

[3]Division of Radiological Physics, Department of Radiology, University of Basel Hospital, Basel, Switzerland

[4]Department of Biomedical Engineering, University of Basel, Allschwil, Switzerland

Correspondence
Nam G. Lee, Alfred E. Mann Department of Biomedical Engineering, University of Southern California, Los Angeles, CA 90089-2564, USA.
Email: namgyunl@usc.edu

Funding information
National Science Foundation, Grant/Award Number: 1828736; National Heart, Lung, and Blood Institute, Grant/Award Number: R01-HL130494

Abstract

Purpose: The reproducibility of scientific reports is crucial to advancing human knowledge. This paper is a summary of our experience in replicating a balanced SSFP half-radial dual-echo imaging technique (bSTAR) using open-source frameworks as a response to the 2023 ISMRM "repeat it with me" Challenge.

Methods: We replicated the bSTAR technique for thoracic imaging at 0.55T. The bSTAR pulse sequence is implemented in Pulseq, a vendor neutral open-source rapid sequence prototyping environment. Image reconstruction is performed with the open-source Berkeley Advanced Reconstruction Toolbox (BART). The replication of bSTAR, termed open-source bSTAR, is tested by replicating several figures from the published literature. Original bSTAR, using the pulse sequence and image reconstruction developed by the original authors, and open-source bSTAR, with pulse sequence and image reconstruction developed in this work, were performed in healthy volunteers.

Results: Both echo images obtained from open-source bSTAR contain no visible artifacts and show identical spatial resolution and image quality to those in the published literature. A direct head-to-head comparison between open-source bSTAR and original bSTAR on a healthy volunteer indicates that open-source bSTAR provides adequate SNR, spatial resolution, level of artifacts, and conspicuity of pulmonary vessels comparable to original bSTAR.

Conclusion: We have successfully replicated bSTAR lung imaging at 0.55T using two open-source frameworks. Full replication of a research method solely relying on information on a research paper is unfortunately rare in research, but our success gives greater confidence that a research methodology can be indeed replicated as described.

KEYWORDS

lung MRI, open-source, replication, reproducible research

1 | INTRODUCTION

The reproducibility of scientific reports is crucial to advancing human knowledge. Due to its growing importance in computational research, the ISMRM has hosted two past reproducible research challenges (organized by the Reproducible Research Study Group in 2019[1] and 2020[2]) and has a current 2023 ISMRM Challenge titled "Repeat it With Me: Reproducibility Team Challenge."[3] This paper is a summary of our experience in replicating bSTAR imaging[4,5] using open-source frameworks as a response to the 2023 ISMRM Challenge.

The bSTAR technique has been proposed for non-electrocardiogram (ECG)-triggered breath-held[4] and free-breathing[5] thoracic imaging with an extremely short TR at 1.5T and has been recently demonstrated at 0.55T.[6] The bSTAR sequence consists of a 3D half-radial dual-echo balanced SSFP (bSSFP) readout in combination with a non-selective hard RF pulse to achieve minimal TR. To minimize eddy currents caused by large jumps in k-space, the bSTAR technique employs smooth k-space trajectories including an Archimedean spiral pattern,[4,7,8] a wobbling Archimedean spiral pole (WASP) pattern,[5] and an adapted spiral phyllotaxis (SP) pattern.[5,9,10]

The bSTAR technique is suitable for various applications at low field strengths. For example, it is especially attractive for lung parenchyma imaging at low field due to prolonged transverse relaxation times.[11] The half-radial bSSFP readout provides an advantage over a full-radial bSSFP readout in reducing concomitant fields, which are inversely proportional to field strength (B_0) and scale quadratically with gradient amplitude. For a trapezoid lobe designed with a fixed maximum gradient amplitude, the accumulated concomitant field phase is proportional to the duration of a trapezoid lobe[12] ; thus, a shorter readout that achieves the same spatial resolution is advantageous at low field. This versatility of the bSTAR technique provides a strong motivation to USC group (N.G.L. and K.S.N.) to initiate this replication study.

In this work, we replicate breath-held bSTAR and self-gated free-breathing bSTAR for thoracic imaging at 0.55T using vendor neutral open-source frameworks to enable code sharing across different institutions, vendors, and scanner software versions. We used the Pulseq framework[13] for pulse sequence implementation, and Berkeley Advanced Reconstruction Toolbox (BART) for image reconstruction.[14] We demonstrate our implementation of the bSTAR sequence on a Siemens hardware platform (Numaris4 VE11S) and interested readers can adapt this implementation to different vendors or platforms. The open-source implementation of bSTAR imaging using Pulseq and BART, denoted open-source bSTAR, is compared to the original author's bSTAR imaging, denoted original bSTAR.

This paper is written in a tutorial style, inspired by tutorial papers in liquid-state NMR.[15–18] We believe this is the best way to describe details to researchers who are less familiar with pulse sequence programming. Since readers may be new to the Pulseq framework, essential basics of Pulseq are introduced as well.

2 | METHODS

2.1 | Pulse sequence

The bSTAR sequence consists of a non-selective hard RF pulse, one bipolar gradient on each gradient axis, one analog-to-digital converter (ADC) window, and hardware time delays. A detailed pulse sequence diagram implemented in the Pulseq framework is illustrated in Figure 1. Implementation details are in Appendix A.

A free-running non-ECG-triggered bSTAR sequence was implemented with three different sampling patterns that provide interleaved smooth k-space trajectories: an Archimedean spiral pattern,[4] a WASP pattern,[5] and an SP pattern.[5] Each 3D trajectory in a sampling pattern is parameterized by a pair of azimuthal angle (ϕ) and polar angle (θ) in the spherical coordinate system.

The phase encoding, readout, and slice directions (denoted PE, RO, and SL, respectively) define the three axes of a right-handed logical coordinate system. The polar angle is measured from the RO direction to the PE-SL plane and the azimuthal angle is measured from the SL direction to the PE-RO plane. Rotating a gradient along the RO direction into any arbitrary direction is achieved by multiplying two right-handed rotation matrices sequentially: (1) apply a rotation matrix that rotates a vector about the PE direction by θ (polar angle), and (2) apply a rotation matrix that rotates a vector about the RO direction by ϕ (azimuthal angle). A graphical illustration of two steps and the definitions of right-handed rotation matrices are shown in Figure 2.

Pulseq provides a Cartesian coordinate system, referred to as Pulseq logical x, y, and z axes. We interpret Pulseq logical axes ("x," "y," "z") as vendor's logical axes (RO, PE, SL) using the "old/compat" option in the "OrientationMapping" parameter of a Pulseq interpreter to comply with the vendor's coordinate transformation from the logical coordinate system (PE, RO, SL) to the physical coordinate system (X, Y, Z).

2.2 | Imaging system

All imaging experiments were performed on a whole-body 0.55T scanner (prototype MAGNETOM Aera; Siemens

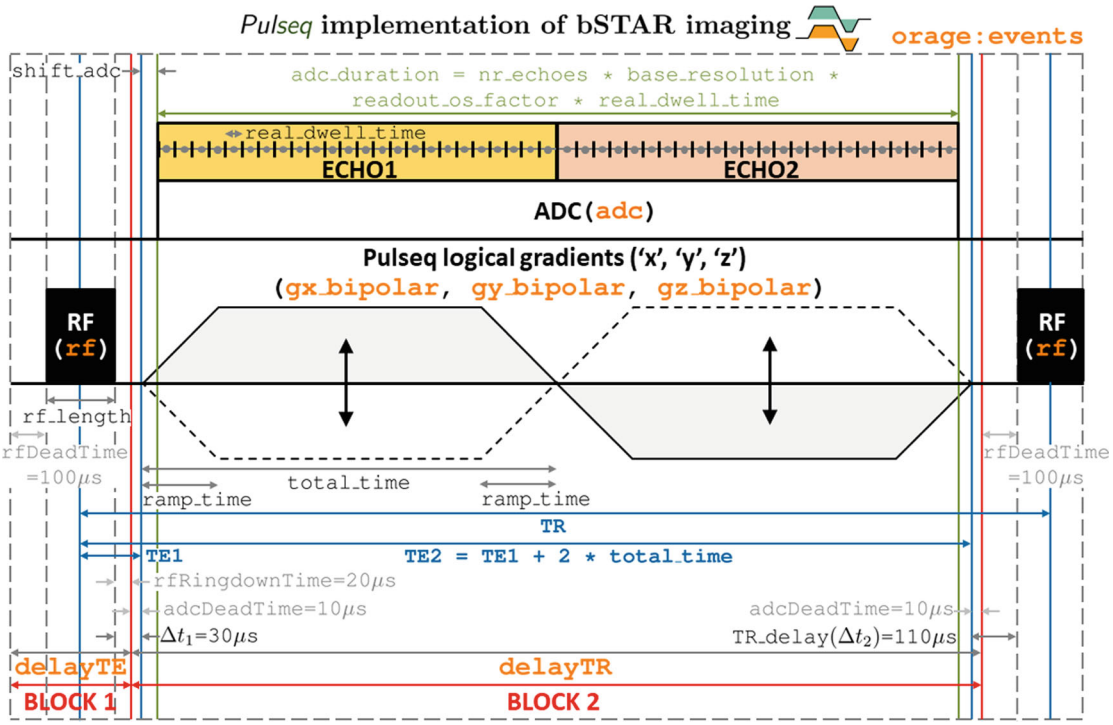

FIGURE 1 Detailed pulse sequence diagram for bSTAR imaging implemented in the open-source Pulseq framework. The bSTAR sequence consists of two sequence blocks (each comprising different events): Block 1 (`rf`, and `delayTE`) and Block 2 (`adc`, `gx_bipolar`, `gy_bipolar`, `gz_bipolar`, and `delayTR`). User-defined parameters are `rf_length`, `base_resolution`, `bandwidth`, and `gradient_factor`. The duration of an ADC window (`adc_duration`) is calculated based on the dwell time with oversampling (`real_dwell_time`). The shorted ramp time to reach the system's maximum gradient strength using the system's maximum slew rate is rounded to (i.e., to the nearest number greater than or equal to) a multiple of `gradRasterTime` (10 μs) and is set to the ramp time of a trapezoid lobe (`ramp_time`). The maximum gradient amplitude is controlled by `gradient_factor` (e.g., a gradient factor of 1 uses the system's maximum gradient strength). The duration of an ADC window is divided by 2, rounded to a multiple of `gradRasterTime` (10 μs), and set to the duration of a trapezoid lobe (`total_time`). A single, base bipolar gradient event is created by combining a positive trapezoid event with a negative trapezoid event. An ADC event (`adc`) requires a hardware time delay, `adcDeadTime`, at the beginning and at the end of the event. `adcDeadTime` is inserted at the beginning and at the end of a bipolar gradient event since the duration of a bipolar gradient event is greater than or equal to the duration of an ADC window (`adc_duration`). Half the time difference between the duration of a bipolar gradient shape and `adc_duration` is rounded to a multiple of 1 μs (`shift_adc`) and is set to the `delay` field of a bipolar gradient event to place an ADC window symmetric around the center of a bipolar gradient shape. ADC sample points are in the centers of time raster steps, where edges of time raster steps are indicated as short bars in ADC. An RF pulse event (`rf`) requires two (system-specific) hardware time delays: `rfDeadTime` and `rfRingdownTime`.

Healthineers, Erlangen, Germany) with gradients capable of 45 mT/m amplitude and 200 T/m/s slew rate. A six-element body coil (anterior) and six elements from an 18-element spine coil (posterior) were used for signal reception.

2.3 | Trajectory measurements

K-space trajectories along the +X, −X, +Y, −Y, +Z, and −Z physical axes were measured with Duyn's method[19] for original bSTAR (implemented in Siemens's IDEA programming language) and with a recently proposed method by Zhao et al.[20] for open-source bSTAR using a 14-cm diameter spherical ball phantom placed at gradient isocenter. Assuming that the measured k-space trajectory scales linearly with the peak gradient amplitude, arbitrarily oriented 3D radial half-spokes were synthesized by a linear combination of the measured k-space trajectories on three physical axes.[4,21] The measurement and correction of B_0 eddy currents[22] were not performed in this study. Note that we used a different trajectory measurement technique for open-source bSTAR, explained in Appendix B.

2.4 | Imaging parameters for the bSTAR sequence

For both phantom and human experiments, the default vendor-calibrated shim setting (i.e., tune-up mode)

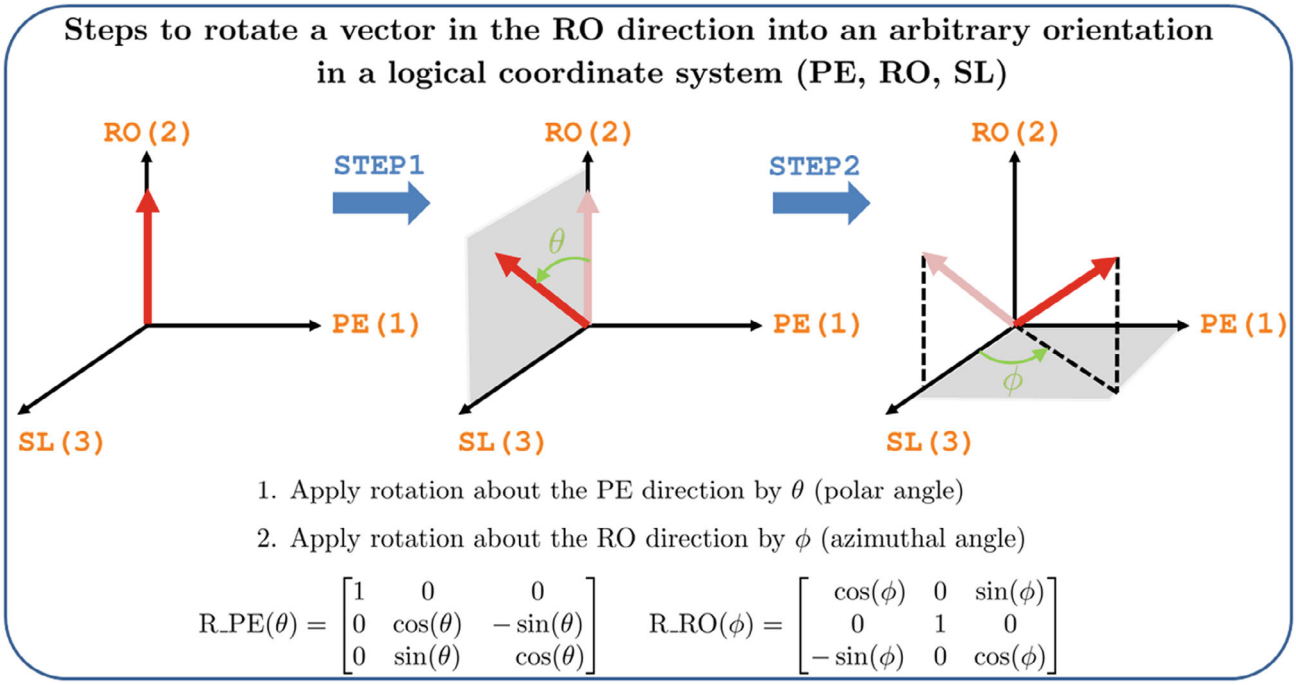

FIGURE 2 Illustration of rotating a vector in the readout (RO) direction into an arbitrary orientation in a right-handed logical coordinate system (PE, RO, SL). All rotation matrices are right-handed. An arbitrary orientation is parameterized by a pair of azimuthal angle (ϕ) and polar angle (θ). A starting vector is always placed on the readout direction. The first step is to apply a rotation matrix that rotates a vector about the PE direction (first direction) by θ. The second step is to apply a rotation matrix that rotates a vector about the RO direction (second direction) by ϕ.

was used with the following imaging parameters: FOV = 34 × 34 × 34 cm, twofold readout oversampling, TE1/TE2/TR = 0.13/1.17/1.38 ms, 200 μs hard RF pulse, flip angle = 25°, bandwidth = 1929 Hz/pixel, 1.61 mm nominal isotropic resolution based on the diameter of k-space coverage, and 288 samples (576 with oversampling) per half-radial (dual-echo) projection. The (α/2-TR/2) preparation[23] followed by a train of 100 dummy TRs with a constant flip angle was used prior to the data acquisition to accelerate the transition into the steady-state. The ramp time and duration of a trapezoid lobe, the duration of an ADC window, and spatial resolution were 140/520/1036.8/1.61, which were perfectly matched with the values displayed in the original bSTAR pulse sequence. The time delay between the end of an RF pulse shape and the beginning of a bipolar gradient shape was 30 μs, which was identical to the time delay Δt_1 (=30 μs) reported in Ref. 5. The time delay between the end of a bipolar gradient shape and the beginning of the next RF pulse shape was 110 μs, which was identical to the time delay Δt_2 (=110 μs) reported in Ref. 5 and identical to the value shown in the original bSTAR pulse sequence (`TR_ delay`). The following parameters were selected in the Siemens Pulseq interpreter: Imaging plane at isocenter, sagittal orientation (A ≫ P) to place the RO direction along the SI direction, 2D mode, and "old/compat" option under "OrientationMapping."

2.5 | Phantom experiments

An accredited American College of Radiology (ACR) structural phantom[24] was scanned with WASP and SP patterns to reproduce Figure 3 of Ref. 4 and Figure 4 of Ref. 5. Four different trajectory patterns were used: WASP patterns using 31 152 half-radial projections with 88 interleaves and 31 150 half-radial projections with 89 interleaves, SP patterns using 31 152 half-radial projections with 88 interleaves and 31 150 half-radial projections with 89 interleaves. The scan time was 43 s.

2.6 | Human experiments

Three healthy volunteers (one male and two females) were scanned under a protocol approved by our institutional review board after providing written informed consent. Both breath-held original bSTAR and open-source bSTAR imaging during end-expiration were performed in a back-to-back manner for two volunteers and in a

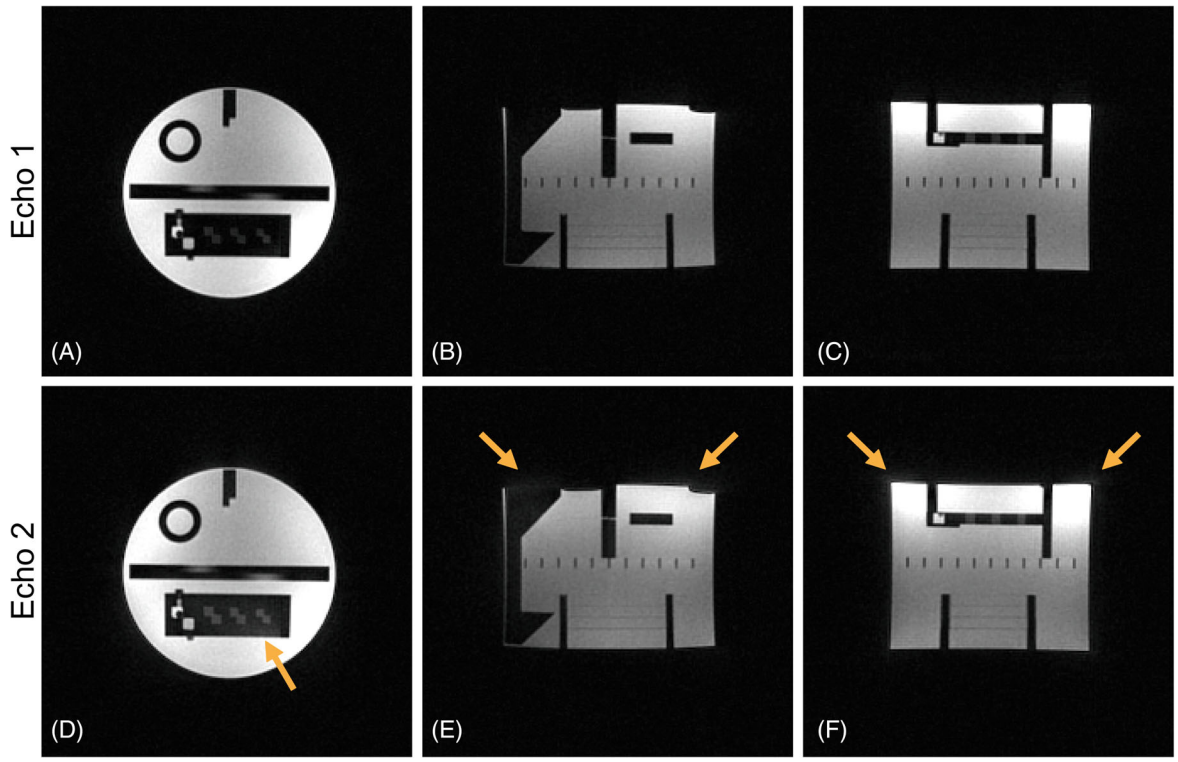

FIGURE 3 Replication of Figure 3 of Ref. 4 using an ACR phantom. Coronal (A and D), sagittal (B and E), and axial (C and F) views are shown for the first echo (top row) and the second echo (bottom row). The spiral phyllotaxis pattern using 31 150 half-radial projections with 89 interleaves was used to minimize eddy currents. The images reconstructed from the second echo show subtle, but enhanced artifacts (e.g., smearing artifacts near the boundaries) compared to those reconstructed from the first echo (orange arrows).

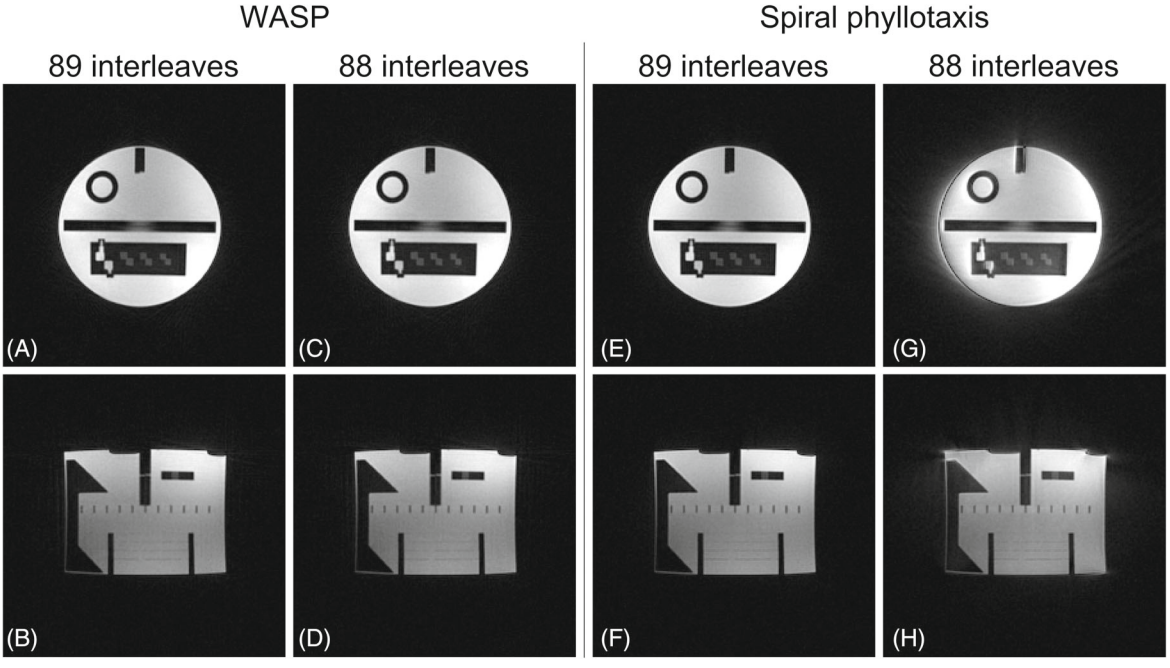

FIGURE 4 Replication of Figure 4 of Ref. 5 using an ACR phantom. A comparison between bSSFP images acquired with bSTAR with WASP patterns and bSTAR with SP patterns. Coronal (A, C, E, G) and sagittal (B, D, F, H) views of echo combined images are shown. Both WASP patterns with 89 and 88 interleaves did not create noticeable eddy current artifacts. Since a SP pattern with a non-Fibonacci number of interleaves is known to create non-smooth trajectories which consequently cause large eddy currents, images reconstructed from the SP pattern with 88 interleaves contain image artifacts as expected. The eddy current artifacts resemble a geometric distortion caused by trajectory errors.

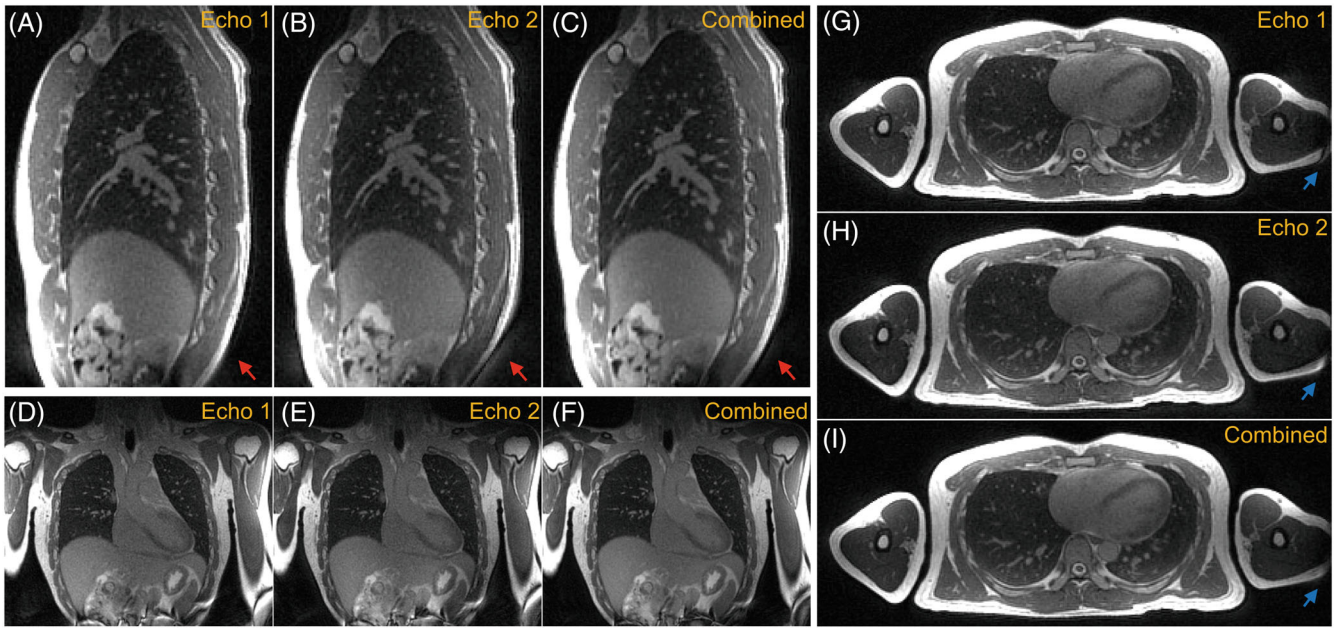

FIGURE 5 Exemplary open-source bSTAR images of volunteer 1 (39/M) acquired during one 50-s end-expiratory breath-hold. Sagittal (A), coronal (D), and axial (G) views of images reconstructed from the first echo (Echo 1). Sagittal (B), coronal (E), and axial (H) views of images reconstructed from the second echo (Echo 2). Sagittal (C), coronal (F), and axial (I) views of echo combined images (Combined). Images reconstructed from the second echo show noticeable artifacts (red arrows), which could be attributed to eddy currents created during the acquisition of the first echo. Banding artifacts (blue arrows) are located far away from the FOV of interest (thoracic area). Echo combined images show improved SNR with less noticeable geometric distortion contributed by the second echo. A movie that pans through all slices is provided in Video S1.

different day for one volunteer. Different trajectory patterns were used: (1) volunteer 1 (39/M) and volunteer 2 (27/F): breath-hold duration = 23.5 s and WASP pattern with 17 000 half-radial projections with 4 interleaves, (2) volunteer 3 (26/F): breath-hold duration = 20.3 s and WASP pattern with 15 000 half-radial projections with 4 interleaves. For volunteer 1, 50-s breath-held open-source bSTAR imaging during end-expiration was additionally performed with the SP pattern using 39 961 half-radial projections with 89 interleaves to assess the image quality of bSTAR without radial undersampling artifacts. A direct comparison between open-source bSTAR and original bSTAR was demonstrated only on volunteer 1 because original bSTAR has applied spatial filtering on reconstructed images except volunteer 1.

2.7 | Image reconstruction

Raw data were converted from vendor proprietary format to the ISMRMRD format[25] and read in MATLAB R2022b (MathWorks, Natick, MA). BART commands were called within MATLAB on a laptop PC equipped with one 2.30 GHz eight-core Intel i7-11800H processor and 128 GB of random-access memory.

Coil sensitivity maps (CSM) were estimated using the following steps: (1) low-resolution CSMs ($32 \times 32 \times 32$) were estimated with nonlinear inversion reconstruction (NLINV)[26] using the `nlinv` command of the BART toolbox ($a = 16$, $b = 16$, 25 Newton steps); (2) low-resolution CSMs were transformed into k-space; (3) zero-padded to a 2X grid (i.e., $2N_1 \times 2N_2 \times 2N_3$) in k-space to account for twofold oversampling in the `nlinv` command; (4) transformed back to image space; (5) cropped from a 2X grid to a 1X grid (i.e., $N_1 \times N_2 \times N_3$) in image space; and (6) each voxel in CSMs was normalized to one.

Image reconstruction was performed with parallel imaging and compressed sensing with ℓ_1-wavelet regularization using the `pics` command of the BART toolbox. Density compensation factors were computed with Ref. 27. We utilized the parameterized fast iterative shrinkage thresholding algorithm (Para-FISTA)[28] which reduces the oscillatory behavior of the fast iterative shrinkage thresholding algorithm (FISTA)[29] during the convergence. This was implemented with modifications to the BART source code. Each echo dataset was reconstructed separately, and complex images from two echoes were added to result in echo combined images. The reconstruction took ~60 min for 30 Para-FISTA iterations with a fixed regularization parameter of 1e-4.

A parameter called `recon_interp_factor` was defined to control reconstruction resolution. This was achieved by scaling normalized k-space trajectories ([−0.5, 0.5] * size of each dimension) by `traj_scale_factor`:

```
traj_scale_factor = ceil(recon_
  matrix_size/recon_interp_factor);
```

All datasets were reconstructed with `recon_matrix_size = [360 360 360]` and `recon_interp_factor = 1.06`, resulting in an interpolated reconstruction resolution of 1.52 mm from a nominal resolution of 1.61 mm.

3 | RESULTS

Figure 3 replicates Figure 3 of Ref. 4 using an ACR phantom. The SP pattern using 31 150 half-radial projections with 89 interleaves was used as opposed to an Archimedean spiral pattern using 18 000 half-radial projections (unclear about the number of interleaves) used in Ref. 4. Both echo images obtained from open-source bSTAR contain no visible artifacts such as off-resonance or banding artifacts and show identical spatial resolution and image quality to those shown in Ref. 4. Note that geometric distortions are not matched due to the use of different gradient sets and image shading is different due to the use of different receive coils. A close inspection of the images reconstructed from the second echo reveals smearing artifacts near the boundaries of the ACR phantom, indicating residual trajectory inaccuracies.

Figure 4 replicates Figure 4 of Ref. 5 using an ACR phantom. Echo combined images are shown. WASP patterns with 88 and 89 interleaves did not create noticeable eddy current artifacts, demonstrating its flexibility in the design of 3D radial trajectory patterns. The SP pattern with the number of interleaves equal to one of the Fibonacci sequence (e.g., 89) provides good image quality as expected. However, when a non-Fibonacci number of interleaves is selected (e.g., 88), the SP pattern becomes non-smooth and severely degrades image quality due to eddy currents caused by large jumps in k-space. The image artifacts due to eddy currents at this scanner (0.55T prototype MAGNETOM Aera, Siemens Healthineers) are not identical to those shown in Ref. 5 (1.5T MAGNETOM Avanto-Fit, Siemens Healthineers). The eddy current artifacts at this scanner rather resemble a geometric distortion caused by trajectory errors.

Figure 5 and Video S1 show the exemplary bSTAR images of volunteer 1 acquired during a single 50-s end-expiratory breath-hold. Images generated using data from the first echo and second echo as well as echo combined images are shown. Images reconstructed from the second echo show noticeable artifacts along the AP direction (i.e., Y physical axis), which could be attributed to B_0 eddy currents generated during the acquisition of the first echo. Banding artifacts are not visible within the FOV of interest (thoracic area). Echo combined images show improved SNR with reduced geometric distortion.

Figure 6 and Video S2 compare echo combined images of volunteer 1 acquired on day 1 with open-source bSTAR against images of volunteer 1 acquired on day 2 with original bSTAR. Each technique acquired data separately with its own pulse sequence (Pulseq vs. IDEA) and performed image reconstruction with different methods for selecting a regularization parameter (fixed regularization vs. data-driven Bayesian shrinkage). Density compensation factors were estimated by Ref. 30 for open-source bSTAR and by the Voronoi method[31] for original bSTAR. A CSM estimation strategy could be different as well. Note that images are not perfectly registered because each dataset was acquired on a different day. Different slices showing similar pulmonary vasculature were selected. Despite the differences in methodology, open-source bSTAR provides adequate SNR, spatial resolution, level of artifacts, and conspicuity of pulmonary vessels comparable to original bSTAR.

4 | DISCUSSION

We have successfully replicated bSTAR lung imaging at 0.55T using two open-source frameworks: (1) Pulseq for a pulse sequence and (2) the BART toolbox for image reconstruction. Full replication of a research method solely relying on information described in research papers is unfortunately rare in research, but our success gives greater confidence that a research methodology can be indeed replicated as described.

An immediate question that one may ask is "would it have been possible without direct interaction/collaboration with original authors?". In an ideal world, the answer would be yes, and all publications would include enough detail to enable "arms-length" replication. However, in this study, we found that there are some important details that require interaction for full replication, such as the specific image filtering method, and the respiratory signal estimation method for self-gated free-breathing bSTAR using the WASP pattern. Even if full details are provided, the replicating group must have sufficient expertise in pulse sequence design and image reconstruction to faithfully reproduce the advertised methodology. Interaction with the original authors is of course necessary when making head-to-head comparisons with the original authors implementation.

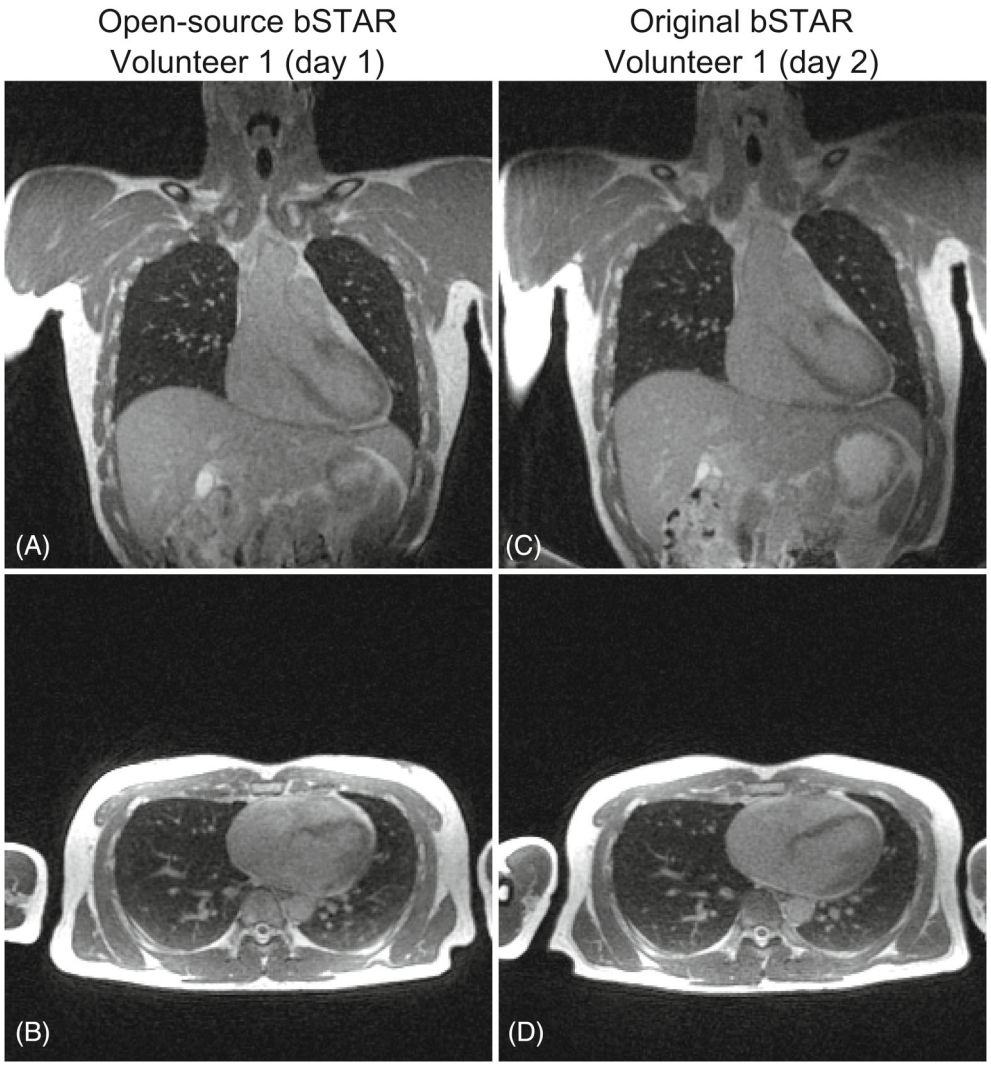

FIGURE 6 Head-to-head comparison between open-source bSTAR and original bSTAR. Coronal (A and C) and axial (B and D) views are shown for open-source bSTAR (left column) and original bSTAR (right column). Note that images are not registered because a dataset for each technique was acquired on a different day from the same volunteer (volunteer 1, 39/M). Each technique acquired a dataset using its own pulse sequence (Pulseq vs. IDEA) and performed image reconstruction with its own reconstruction pipeline. Despite the differences in methodology, open-source bSTAR provides image quality comparable to original bSTAR. No spatial filtering was applied on images from both techniques. A movie that pans through only coronal slices is provided in Video S2.

We found Pulseq to be an excellent open-source framework for fast prototyping a pulse sequence. However, it is important to realize that images acquired with pulse sequences written in Pulseq are not identical to those acquired with vendor product pulse sequences or those written in a vendor provided programming environment. They become close to each other (not 100% identical considering subtle differences in image reconstruction) only when all details in pulse sequence design are perfectly matched. For bSTAR imaging, it was very fortunate that the bSSFP kernel was very simple to implement. One limitation of Pulseq is that Pulseq does not provide full capabilities that vendor proprietary programming languages can provide. For example, adding an additional user-selectable box to locate an inversion pulse is not possible without modifying the Pulseq interpreter source code substantially, which is beyond the capability of normal Pulseq users.

The BART toolbox provides a rich set of MRI reconstruction algorithms. We greatly benefit from BART's FISTA implementation when modifying FISTA to Para-FISTA. Without BART, developing a reconstruction algorithm from scratch in a middle-level language (e.g., C/C++) would have been a daunting task. It is important to note that having basic knowledge of C/C++ to comprehend the BART source code and mathematical skills (e.g., convex optimization) are essential when implementing new algorithms in BART. Although the BART toolbox is a great open-source framework for fast prototyping new reconstruction algorithms, the pics command currently supports only single GPU (version 8.0.0). Its limited multi-GPU support may limit widespread use of BART among researchers handling multi-dimensional non-Cartesian datasets.

There are several limitations of this work: (1) both acquisition and image reconstruction were implemented in open-source frameworks, but open-source bSTAR to date has only been tested at one center; and (2) open-source bSTAR has been demonstrated only with limited sample size. We plan to address these limitations in future studies.

5 | CONCLUSIONS

We have successfully replicated the bSTAR technique using open-source frameworks, and replicated figures shown in the published literature[4,5] with comparable quality. This study also demonstrates the power of open-source frameworks, especially Pulseq, because designing a pulse sequence in a vendor proprietary environment requires expertise and tremendous effort.

ACKNOWLEDGMENTS

We acknowledge grant support from the National Science Foundation (#1828736) and research support from Siemens Healthineers.

FUNDING INFORMATION

National Science Foundation #1828736; NIH R01-HL130494.

DATA AVAILABILITY STATEMENT

The code and data that reproduce the results of this study are openly available in Github at https://www.github.com/usc-mrel/replication_bstar.

ORCID

Nam G. Lee https://orcid.org/0000-0001-5462-1492
Oliver Bieri https://orcid.org/0000-0002-6755-5495
Krishna S. Nayak https://orcid.org/0000-0001-5735-3550

REFERENCES

1. Can you reproduce this seminal MRM paper? Participate in the reproducible research study group challenge! - ISMRM's MR Pulse Blog. https://blog.ismrm.org/2019/04/02/ismrm-reproducible-research-study-group-2019-reproduce-a-seminal-paper-initiative/. Accessed June 15, 2023
2. Reproducibility Challenge 2020: Join the reproducible research and quantitative MR study groups in their efforts to standardize T1 mapping – ISMRM's MR Pulse Blog. https://blog.ismrm.org/2019/12/12/reproducibility-challenge-2020-join-the-reproducible-research-and-quantitative-mr-study-groups-in-their-efforts-to-standardize-t1-mapping/. Accessed June 15, 2023
3. 2023–24 Reproducibility Challenge – ISMRM Challenge. https://challenge.ismrm.org/2023-24-reproducibility-challenge/. Accessed June 15, 2023
4. Bauman G, Bieri O. Balanced steady-state free precession thoracic imaging with half-radial dual-echo readout on smoothly interleaved archimedean spirals. *Magn Reson Med*. 2020;84:237-246. doi:10.1002/MRM.28119
5. Bieri O, Pusterla O, Bauman G. Free-breathing half-radial dual-echo balanced steady-state free precession thoracic imaging with wobbling Archimedean spiral pole trajectories. *Z Med Phys*. 2023;33:220-229. doi:10.1016/J.ZEMEDI.2022.01.003
6. Bauman G, Lee NG, Tian Y, Bieri O, Nayak KS. Submillimeter lung MRI at 0.55 T using balanced steady-state free precession with half-radial dual-echo readout (bSTAR). *Magn Reson Med*. 2023;90:1949-1957. doi:10.1002/MRM.29757
7. Wong STS, Roos MS. A strategy for sampling on a sphere applied to 3D selective RF pulse design. *Magn Reson Med*. 1994;32:778-784. doi:10.1002/MRM.1910320614
8. Nielles-Vallespin S, Speier P, Bi X, Li D, Mueller E. Navigator-gated whole heart coronary MRI: Comparison of 3D TrueFisp cartesian and radial acquisitions. Proc Intl Soc Mag Reson Med. 2006. p 366.
9. Piccini D, Littmann A, Nielles-Vallespin S, Zenge MO. Spiral phyllotaxis: the natural way to construct a 3D radial trajectory in MRI. *Magn Reson Med*. 2011;66:1049-1056. doi:10.1002/MRM.22898
10. Delacoste J, Chaptinel J, Beigelman-Aubry C, Piccini D, Sauty A, Stuber M. A double echo ultra short echo time (UTE) acquisition for respiratory motion-suppressed high resolution imaging of the lung. *Magn Reson Med*. 2018;79:2297-2305. doi:10.1002/MRM.26891
11. Li B, Lee NG, Cui SX, Nayak KS. Lung parenchyma transverse relaxation rates at 0.55 T. *Magn Reson Med*. 2023;89:1522-1530. doi:10.1002/MRM.29541
12. Bernstein MA, Zhou XJ, Polzin JA, et al. Concomitant gradient terms in phase contrast MR: analysis and correction. *Magn Reson Med*. 1998;39:300-308. doi:10.1002/MRM.1910390218
13. Layton KJ, Kroboth S, Jia F, et al. Pulseq: a rapid and hardware-independent pulse sequence prototyping framework. *Magn Reson Med*. 2017;77:1544-1552. doi:10.1002/MRM.26235
14. Tamir JI, Ong F, Cheng JY, Uecker M, Lustig M. Generalized magnetic resonance image reconstruction using the berkeley advanced reconstruction toolbox. doi:10.5281/zenodo.31907
15. Kuprov I. Large-scale NMR simulations in liquid state: A tutorial. *Magn Reson Chem*. 2018;56:415-437. doi:10.1002/MRC.4660
16. Concilio MG. Large-scale magnetic resonance simulations: A tutorial. *Magn Reson Chem*. 2020;58:691-717. doi:10.1002/MRC.5018
17. Bengs C, Levitt MH. SpinDynamica: symbolic and numerical magnetic resonance in a Mathematica environment. *Magn Reson Chem*. 2018;56:374-414. doi:10.1002/MRC.4642
18. Jeannerat D. Software tools and tutorials in liquid state NMR. *Magn Reson Chem*. 2018;56:373-373. doi:10.1002/MRC.4734
19. Duyn JH, Yang Y, Frank JA, van der Veen JW. Simple correction method for k-space trajectory deviations in MRI. *J Magn Reson*. 1998;132:150-153. doi:10.1006/JMRE.1998.1396
20. Zhao X, Lee H, Song HK, Cheng CC, Wehrli FW. Impact of gradient imperfections on bone water quantification with UTE MRI. *Magn Reson Med*. 2020;84:2034-2047. doi:10.1002/MRM.28272
21. Lu A, Brodsky E, Grist TM, Block WF. Rapid fat-suppressed isotropic steady-state free precession imaging using true 3D multiple-half-echo projection reconstruction. *Magn Reson Med*. 2005;53:692-699. doi:10.1002/MRM.20389
22. Brodsky EK, Klaers JL, Samsonov AA, Kijowski R, Block WF. Rapid measurement and correction of phase errors from B0 eddy currents: impact on image quality for non-cartesian imaging. *Magn Reson Med*. 2013;69:509-515. doi:10.1002/MRM.24264
23. Deiming M, Heid O. Magnetization prepared true FISP imaging. 1994 https://www.researchgate.net/publication/308954649. Accessed June 20, 2023

24. Ihalainen TM, Lönnroth NT, Peltonen JI, et al. MRI quality assurance using the ACR phantom in a multi-unit imaging center. *Acta Oncol*. 2011;50:966-972. doi:10.3109/0284186X.2011.582515
25. Inati SJ, Naegele JD, Zwart NR, et al. ISMRM raw data format: a proposed standard for MRI raw datasets. *Magn Reson Med*. 2017;77:411-421. doi:10.1002/MRM.26089
26. Uecker M, Hohage T, Block KT, Frahm J. Image reconstruction by regularized nonlinear inversion—joint estimation of coil sensitivities and image content. *Magn Reson Med*. 2008;60:674-682. doi:10.1002/MRM.21691
27. ISMRM/mri_unbound: Code from the mri_unbound website. https://github.com/ISMRM/mri_unbound. Accessed June 18, 2023
28. Liang J, Luo T, Schonlieb CB. Improving "Fast Iterative Shrinkage-Thresholding Algorithm": faster, smarter, and greedier. *SIAM J Sci Comput*. 2022;44:A1069-A1091. doi:10.1137/21M1395685
29. Beck A, Teboulle M. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM J Imag Sci*. 2009;2:183-202. doi:10.1137/080716542
30. Zwart NR, Johnson KO, Pipe JG. Efficient sample density estimation by combining gridding and an optimized kernel. *Magn Reson Med*. 2012;67:701-710. doi:10.1002/MRM.23041
31. Rasche V, Proksa R, Sinkus R, Börnert P, Eggers H. Resampling of data between arbitrary grids using convolution interpolation. *IEEE Trans Med Imaging*. 1999;18:385-392. doi:10.1109/42.774166
32. Herz K, Mueller S, Perlman O, et al. Pulseq-CEST: towards multi-site multi-vendor compatibility and reproducibility of CEST experiments using an open-source sequence standard. *Magn Reson Med*. 2021;86:1845-1858. doi:10.1002/MRM.28825

SUPPORTING INFORMATION

Additional supporting information may be found in the online version of the article at the publisher's website.

Video S1. Animation showing exemplary open-source bSTAR images of volunteer 1 (39/M) acquired during one 50-s end-expiratory breath-hold.

Video S2. Animation showing coronal slices of the head-to-head comparison between open-source bSTAR and original bSTAR.

How to cite this article: Lee NG, Bauman G, Bieri O, Nayak KS. Replication of the bSTAR sequence and open-source implementation. *Magn Reson Med*. 2024;91:1464-1477. doi: 10.1002/mrm.29947

APPENDIX A

A.1 Pulseq introduction

The Pulseq project provides: (1) a file specification to create a human-readable text file containing low-level sequence instructions, known as a Pulseq sequence file (herein referred to as *pulseq-file*[32]); (2) high-level sequence design tools written in programming languages such as MATLAB (The MathWorks, Natick, MA) and Python; and (3) a vendor-specific interpreter module that translates the content of a *pulseq-file* to vendor-specific hardware commands. This interpreter module is also referred to as a Pulseq interpreter sequence or a Pulseq interpreter and is written in vendor's proprietary pulse programming framework. In this work, a *pulseq-file* is created with custom MATLAB code and Pulseq toolbox functions (mr toolbox) provided with the MATLAB Pulseq package (version higher than 1.4.0).

The Pulseq framework provides RF, gradient, ADC, and delay events as basic constitutes for building a pulse sequence. In Pulseq, a pulse sequence is comprised of so-called sequence blocks. A sequence block can be comprised of a single event or a combination of events. A gradient event defined on a different direction/axis is considered as a distinct event. It is important to emphasize that only one gradient event per axis can be added to a sequence block. A *pulseq-file* can contain an arbitrary number of sequence blocks.

A.2 Pulseq implementation of the bSTAR sequence

The bSTAR sequence is comprised of a non-selective hard RF pulse, a bipolar gradient, one analog-to-digital converter (ADC) window, and hardware time delays. In Pulseq, the bSTAR sequence consists of two sequence blocks: Block 1 and Block 2. Block 1 consists of one RF event (`rf`) and one delay event (`delayTE`). Block 2 consists of three gradient events (`gx`, `gy`, `gz`), one ADC event (adc), and one delay event (`delayTR`). The design of each event is described in the following subsections.

Events of arbitrary type such as shaped RF pulses and arbitrary gradients contain discrete samples of a continuous waveform. To create discrete samples, a sampling grid consisting of time steps is defined and used to evaluate an analytic expression of a continuous waveform. Depending on vendor's convention, a sampling point can be located either at the center, left edge, or right edge of a time step. To comply with Siemen's convention, sampling points are located at the centers of time steps.

For RF samples, gradient samples, and ADC samples, each has its minimum raster time, known as `rfRasterTime`, `gradRasterTime`, `adcRasterTime`, respectively. The dwell time of a time step can be an integer multiple of its minimum dwell time.

It is important to understand the concept of time raster alignment defined in Pulseq. An event can start and end only at time points which are multiples of its minimum raster time. Similarly, a sequence block can

start and end only at time points that are multiples of `blockDurationRaster` (e.g., 10 μs for Siemens).

Hardware time delays are imposed on RF and ADC events. For RF events, two hardware time delays are required. The one imposed at the beginning of an RF event is known as `rfDeadTime`, and the other imposed at the end of an RF event is known as `rfRingdownTime`. For ADC events, a hardware time delay, known as `adcDeadTime`, is imposed on both ends of the ADC event. These delays are vendor and hardware platform dependent. Specifically, a non-zero value of `adcDeadTime` is required for Siemens (e.g., 10 μs), and for other vendors the sequence designer might need to use different dead times before and after the ADC event. Note that the required hardware time delays only apply to RF and ADC events. Events of different type can be played during the period of a required hardware time delay.

A.2.1. System specification (`sys`)

The data structure `sys` contains the specification about scanner hardware and is defined by calling the `opts.m` function:

```
sys = mr.opts('MaxGrad', max_grad,
   'GradUnit', 'mT/m',...
'MaxSlew', max_slew, 'SlewUnit',
   'T/m/s',...
'rfRingdownTime', 20e-6,...
'rfDeadTime', 100e-6,...
'adcDeadTime', 10e-6,...
'B0', B0);
```

where the "`MaxGrad`" field stores the maximum gradient strength in Hz/m, which is obtained by multiplying `max_grad` in mT/m with `sys.gamma` in Hz/T, and the "`MaxSlew`" field stores the maximum slew rate in Hz/m/s, which is obtained by multiplying `max_slew` in T/m/s with `sys.gamma` in Hz/T. The "`B0`" field stores the main field strength in Tesla. The data structure sys contains the following additional fields:

```
        adcRasterTime: 1.0000e-07
         rfRasterTime: 1.0000e-06
       gradRasterTime: 1.0000e-05
  blockDurationRaster: 1.0000e-05
                gamma: 42576000
```

A.2.2. Block 1: RF event (`rf`)

An α-degree non-selective hard RF pulse (`rf`) is created by calling the `makeBlockPulse.m` function:

```
rf = mr.makeBlockPulse(flip_angle
* pi / 180, sys, 'Duration', rf_length);
```

where `flip_angle` is the flip angle of a hard RF pulse in radians, `rf_length` is the duration of a hard RF pulse in seconds. The data structure `rf` contains the following fields:

```
              rf =
            type: 'rf'
          signal: [347.2222 347.2222]
               t: [0 2.0000e-04]
       shape_dur: 2.0000e-04
      freqOffset: 0
     phaseOffset: 0
        deadTime: 1.0000e-04
    ringdownTime: 2.0000e-05
           delay: 1.0000e-04
```

where `signal` contains RF samples in Hertz, `t` contains time samples in seconds, `shape_dur` is the duration of an RF pulse shape in seconds, `freqOffset` is the frequency offset of an RF pulse in Hertz, `phaseOffset` is the phase offset of an RF pulse in radians, `deadTime` is the hardware time delay required at the beginning of an RF pulse in seconds, `ringdownTime` is the hardware time delay required after the end of an RF pulse in seconds, and `delay` is the delay before starting the RF pulse shape in seconds, which is greater than or equal to `deadTime`.

A.2.3. Block 1: Delay event (`delayTE`)

The block duration is determined by the longest event in a block and each block must start and end at multiples of `sys.blockDurationRaster`. Because some events (e.g., ADC event) have finer raster times than a block, it is easier to conform block time raster alignment using a single delay event that starts and ends at multiples of `sys.blockDurationRaster` while encapsulating all events in each block. Therefore, a single delay event (`delayTE`) is used to encapsulate the RF event (`rf`) and hardware time delays in Block 1. An ADC event requires a hardware time delay of 10 μs (`sys.adcDeadTime`) at the beginning. Since the duration of an ADC window can be equal to or shorter than the duration of a bipolar gradient, this time delay is added to the bipolar gradient for simplicity. The start time of a bipolar gradient is TE1 (first TE), and thus the end time of Block 1 should be 10 μs earlier than TE1. A TE is calculated from the isodelay point of an RF pulse, which is at the center of a hard RF pulse in this case. Therefore, a delay event (`delayTE`) can be calculated as follows:

```
delayTE = round((rf.deadTime +
rf_length / 2 + TE1 - sys.adcDeadTime) /
sys.gradRasterTime) * sys.gradRasterTime;
```

Note that `sys.gradRasterTime` is identical to `sys.blockDurationRaster`. The sum of `sys.rfRingdownTime` and `sys.adcDeadTime` dictates the

minimum TE1 and is referred to as the required hardware time delay Δt_1 between RF and ADC in Ref. 4, 5.

A.2.4. Block 2: Trapezoid gradient design

Open-source bSTAR requires several user-defined parameters to control the spatial resolution: number of samples per echo without readout oversampling (`base_resolution`), bandwidth in Hz per pixel (`bandwidth`), and readout oversampling factor (`readout_os_factor`). To design a trapezoid lobe, the ADC dwell time with oversampling (`real_dwell_time`) is first calculated as follows:

```
real_dwell_time = round((1 / bandwidth) /
 (readout_os_factor * base_resolution) *
  1e7) * 1e-7;
```

Note that the dwell time is rounded down to a multiple of 100 ns, which is the minimum ADC raster time (`sys.adcRasterTime`). The number of ADC samples and the duration of an ADC window are calculated as.

```
adc_samples = nr_echoes *
  base_resolution * readout_os_factor;
adc_duration = adc_samples *
  real_dwell_time;
```

The gradient amplitude of a trapezoid lobe in millitesla per meter is calculated based on FOV:

```
amplitude = 1 / (fov_read *
  readout_os_factor) / (sys.gamma *
  real_dwell_time * 1e-3);
```

The shortest ramp time to reach the system's maximum gradient strength using the system's maximum slew rate is rounded to (i.e., to the nearest number greater than or equal to) a multiple of `gradRasterTime` (e.g., 10 μs) and is set to the ramp time of a trapezoid lobe (`ramp_time`):

```
ramp_time = ceil((sys.maxGrad/sys.
  maxSlew) / sys.gradRasterTime) *
  sys.gradRasterTime;
```

The duration of an ADC window is divided by 2, rounded to a multiple of `gradRasterTime` (e.g., 10 μs), and set to the duration of a trapezoid lobe (`total_time`):

```
total_time = ceil(adc_duration /
  (sys.gradRasterTime * 2)) *
  (sys.gradRasterTime * 2) / 2;
```

By design, the duration of a trapezoid lobe is equal to or greater than the duration of an ADC window.

A.2.5. Block 2: Base bipolar gradient (`g_bipolar`)

Only arbitrary gradient events and trapezoid gradient events are supported in the current version (1.4.1) of a file specification. Thus, a bipolar trapezoid gradient event should be designed as an arbitrary gradient event. In this subsection, we focus on creating the gradient shape of a bipolar gradient event along the readout direction. A simple way to design a bipolar gradient event using the mr toolbox is described as follows. First, a positive trapezoid event (`g_positive`) is created by calling the `makeTrapezoid.m` function:

```
g_positive = mr.makeTrapezoid('x',
   'riseTime', ramp_time,
'flatTime', total_time - 2 *
   ramp_time, 'fallTime', ramp_time,
   'amplitude', sys.gamma * amplitude
   * 1e-3);
```

where `ramp_time` is the rise time and fall time of a trapezoid gradient in seconds, `total_time` is the sum of the rise time, plateau time, and fall time of a trapezoid gradient in seconds, and `amplitude` is the amplitude of a trapezoid gradient in mT/m. The unit of the `amplitude` field is Hz/T and thus the `amplitude` variable is scaled by `sys.gamma`, which is in Hz/T. The data structure `g_positive` contains the following fields:

```
g_positive =
     type: 'trap'
  channel: 'x'
amplitude: 8.1697e+05
 riseTime: 1.4000e-04
 flatTime: 2.4000e-04
 fallTime: 1.4000e-04
     area: 310.4468
 flatArea: 196.0717
    delay: 0
    first: 0
     last: 0
```

where `type` indicates the type of a gradient event ("`trap`" for trapezoid gradients or "`grad`" for arbitrary gradients), `channel` indicates the designated gradient axis ("`x`," "`y`," or "`z`"), amplitude is the amplitude of a gradient event in Hz/m, `area` and `flatArea` are the entire area and plateau area of a trapezoid gradient in Hz/m·sec, respectively, and `delay` is the delay before starting the gradient event in seconds. Second, a negative trapezoid event (`g_negative`) is created by scaling `g_positive` with −1 using the `scaleGrad.m` function and setting its delay to the duration of a positive trapezoid event:

```
g_negative = mr.scaleGrad
  (g_positive,-1);
```

```
g_negative.delay = mr.calcDuration
  (g_positive);
```

where the `calcDuration.m` function returns the entire duration (delay + shape duration) of any event in seconds. Finally, a bipolar gradient event (`g_bipolar`) is created by combining `g_positive` and `g_negative` using the `addGradients.m` function:

```
g_bipolar = mr.addGradients
  ({g_positive, g_negative}, sys);
g_bipolar.delay = sys.adcDeadTime;
```

Note that a bipolar gradient event is delayed because an ADC event requires a time delay of 10 μs (`sys.adcDeadTime`) at the beginning of the event. The data structure `g_bipolar` contains the following fields:

```
g_bipolar =
      type: 'grad'
   channel: 'x'
  waveform: [0 8.1697e+05 8.1697e+05 0
             -8.1697e+05 -8.1697e+05 0]
     delay: 1.00e-05
        tt: [0 1.40e-04 3.80e-04
             5.20e-04 6.60e-04 9.00e-04
             0.00104]
 shape_dur: 0.00104
     first: 0
      last: 0
```

where the type of a bipolar gradient event is arbitrary gradient ('`grad`') and its waveform consists of seven samples. Note that, although this gradient event is of arbitrary type, gradient samples are not located at the centers of time steps.

A.2.6. Block 2: ADC event (`adc`)

We intend to align the center of an ADC event to the center of a bipolar gradient shape (i.e., `waveform`). However, ADC events can only start at time points which are multiples of 1 μs (`sys.adcRasterTime * 10`). To achieve this, the time difference between the duration of a bipolar gradient shape and the duration of ADC samples is calculated. The duration of ADC samples (`adc_duration`) is calculated as the product of the number of ADC samples and the ADC dwell time, which is equal to the elapsed time from the left edge of the first time step to the right edge of the last time step. Then, `shift_adc` is calculated as half of this difference rounded to a multiple of 1 μs (`sys.adcRasterTime * 10`):

```
shift_adc = round((2 * total_time -
   adc_duration) / 2 / (sys.adcRasterTime
   * 10)) * (sys.adcRasterTime * 10);
```

Since the delay of a bipolar gradient event is set to 10 μs to comply with the hardware time delay requirement of an ADC event (`sys.adcDeadTime`), the delay of an ADC event (`adc_delay`) is set to the sum of `sys.adcDeadTime` and `shift_adc`. Finally, an ADC event (`adc`) is created by calling the `makeAdc.m` function:

```
adc_delay = sys.adcDeadTime +
   shift_adc;
adc = mr.makeAdc(adc_samples,
   'Dwell', real_dwell_time, 'delay',
   adc_delay, 'system', sys);
```

A.2.7. Block 2: Delay event (`delayTR`)

Similar to `delayTE`, a single delay event (`delayTR`) encapsulates all events and hardware time delays in Block 2. A hardware time delay of 10 μs (`sys.adcDeadTime`) is appended at the end of a bipolar gradient event to comply with the requirement of an ADC event. Thus, the minimum duration of `delayTR` is the duration of a bipolar gradient event plus two appended hardware time delays. In general, a user can set an arbitrary TR and a delay event (`delayTR`) can be calculated as follows:

```
delayTR = round((TR -
   delayTE) / sys.gradRasterTime) *
   sys.gradRasterTime;
```

Note that the next RF pulse requires a hardware time delay of 100 μs (`sys.rfDeadTime`). The time interval from the end of a bipolar gradient shape to the start time of the next RF pulse shape is the sum of `sys.adcDeadTime` and `sys.rfDeadTime` (e.g., 110 μs). This time interval is denoted as `TR_delay` in this paper and referred to as the required hardware time delay Δt_2 between ADC and RF in Ref. 4,5.

A.2.8. Block 2: Bipolar gradient events (`gx`, `gy`, `gz`)

We use three Cartesian coordinate systems: the logical coordinate system (LCS), the physical coordinate system (PCS), and the Pulseq logical coordinate system. We define the coordinates of the LCS as (PE, RO, SL), which stands for phase encode, readout, and slice directions, respectively. Similarly, we define the coordinates of the PCS as (X, Y, Z) and the coordinates of the Pulseq logical coordinate system as ("x," "y," "z"). Once arbitrary orientated gradients are calculated with right-handed rotation matrices in the LCS (PE, RO, SL), the gradient shapes along the PE, RO, SL directions are assigned to the Pulseq logical "y," "x," and "z" axes, respectively.

APPENDIX B

We used different trajectory measurement techniques for original bSTAR and open-source bSTAR. When using

Duyn's method at 0.55T, a substantial amount of a concomitant field-induced phase could be created at off-center slices when a large gradient amplitude is played along the slice-selection direction. These spatially varying phase errors cause bias in the estimation of measured k-space trajectories. We used the method developed by Zhao et al.[20] because it is more robust to concomitant fields. Specifically, (1) a slice at isocenter is used so that concomitant fields are minimized; and (2) a linear phase slope in the excited slice is estimated with global least-squares fitting using all spatially resolved voxels that are affected by a different amount of concomitant fields. This weighted fitting reduces bias in the estimation of k-space trajectories. After comparing image quality of the second echo images obtained with Duyn's method and Zhao's method (not shown), Zhao's method was chosen because it reduced artifacts along the AP direction.