Published in final edited form as:

Magn Reson Med. 2016 January ; 75(1): 257–265. doi:10.1002/mrm.25640.

Pulse Sequence Programming in a Dynamic Visual Environment: SequenceTree

Jeremy F. Magland, Cheng Li, Michael C. Langham, and Felix W. Wehrli

Department of Radiology, University of Pennsylvania, Philadelphia, PA 19104, USA

Abstract

Purpose—To describe SequenceTree (ST), an open source, integrated software environment for implementing MRI pulse sequences, and ideally exported them to actual MRI scanners. The software is a user-friendly alternative to vendor-supplied pulse sequence design and editing tools and is suited for non-programmers and programmers alike.

Methods—The integrated user interface was programmed using the Qt4/C++ toolkit. As parameters and code are modified, the pulse sequence diagram is automatically updated within the user interface. Several aspects of pulse programming are handled automatically allowing users to focus on higher-level aspects of sequence design. Sequences can be simulated using a built-in Bloch equation solver and then exported for use on a Siemens MRI scanner. Ideally other types of scanners will be supported in the future.

Results—The software has been used for eight years in the authors' laboratory and elsewhere and has been utilized in more than fifty peer-reviewed publications in areas such as cardiovascular imaging, solid state and non-proton NMR, MR elastography, and high resolution structural imaging.

Conclusion—ST is an innovative, open source, visual pulse sequence environment for MRI combining simplicity with flexibility and is ideal for both advanced users and those with limited programming experience.

Keywords

software; pulse sequence; graphical user interface; MRI simulation

INTRODUCTION

Magnetic Resonance Imaging (MRI) is unique in that once the hardware requirements are met, the gateway to new method development is the pulse sequence, or data acquisition technique. Advances in methodology in recent years have led to an explosion of new acquisition techniques with dozens of variations on acquisition strategies presented at scientific meetings and publications each year. Many of these sequences are never reproduced outside the lab of the original implementation due to material transfer issues

and/or incompatibility between scanner types or software versions. The complexity of the programming environment for creation of new pulse sequences can be a major obstacle to implementing new ideas, or reproducing existing methods.

For pulse programming environments, there is generally a trade-off between user-friendliness on the one hand and flexibility on the other. A purely visual environment, for example one in which the programmer intuitively manipulates the sequence using mouse clicks, may lack the flexibility required to design pulse sequences that deviate substantially from common Cartesian or radial sampling schemes. On the other hand, in a code-based development environment flexible enough to cover the widest range of possible research sequences, pulse programs can be difficult to read and time-consuming to develop and debug.

Whereas the vast majority of pulse sequences are programmed using software frameworks provided by scanner vendors, some open source projects have emerged to tackle the problem of scanner-independent pulse programming. One such framework is the Object-oriented Development Interface for NMR (ODIN) (1). ODIN provides a platform-independent set of libraries for developing research pulse sequences using custom C++ classes. It simplifies pulse programming by separating the sequence into a hierarchy of logical units (the so-called sequence tree). However, new features of the sequence must be added through manipulation of the C++ source code, which can be a daunting task for researchers with little object-oriented programming experience. For example, a basic gradient echo pulse sequence programmed in ODIN comprises more than 300 lines of C++ code (<http://od1n.sourceforge.net/>).

Open source software also exists for developing sequences for the purpose of performing MRI simulations using numerical Bloch equation simulators (2,3). For example, the JEMRIS software (2) includes a module for interactively designing arbitrary pulse sequences for use with the simulation module. However, such sequences cannot be run on a physical scanner.

Other open source projects (e.g., Gadgetron (4)) focus on improving reproducibility in medical imaging research by providing scanner-independent platforms for image reconstruction and image post-processing. But these frameworks do not attempt to tackle the challenging problem of creating an open, scanner-independent platform for data acquisition.

The environments provided by scanner vendors are meant to produce clinically-usable sequences, and are not necessarily optimized for research. From the perspective of users developing new techniques (or reproducing techniques from the literature), there is a need for a rapid, proof-of-concept prototyping framework. Ideally this should be a what-you-see-is-what-you-get (WYSIWYG) environment allowing non-programmers to implement these new ideas with minimal coding. Such a tool would also be invaluable for training and education of researchers new to the field.

In this article we describe SequenceTree (ST), an open source, visual pulse sequence programming environment that combines ease-of-use with flexibility and that can be interfaced to any scanner provided that an appropriate conversion tool is created. Thus far

ST has only been interfaced to Siemens scanners (Siemens Medical Solutions, Erlangen, Germany), but the software is designed to be flexible with more than 97% of the code being scanner-independent. The aim is to provide software that is intuitive for basic users (i.e., MRI researchers with little programming experience) as well as for advanced users (i.e., those familiar with the C++ programming language). To enable such a user experience, many aspects of pulse sequence execution, such as phase encoding and field-of-view shifts (5), are handled automatically by the framework. The end result is an integrated user interface allowing the programmer to visualize the sequence output at all stages while visually manipulating sequence elements, modifying parameters or adjusting source code. The software is an extension of a tool (of the same name) introduced by the authors eight years ago (6). In the time since the initial version was created, ST has evolved to meet the needs of researchers in diverse areas of MRI research. In what follows we describe in detail the methodology of the software and highlight a few example applications from the authors' laboratory.

METHODS

ST comprises four modules: (a) a graphical user interface (written in C++/Qt4) showing a pulse sequence diagram which updates automatically as parameters and code are changed; (b) a program for compiling and running sequences; (c) a virtual scanner for simulating sequences on digital phantoms; (d) an interface for communicating with a clinical scanner. In this section we provide design details for these components. More details can be found at the ST website (<http://www.sequencetree.org>).

Pulse sequence structure

A ST pulse sequence consists of a hierarchical tree of nodes (Figure 1) with each node comprising a snippet of C++ code and a collection of parameter values. The singular top-most node (the *root node*) contains parameters that affect the entire sequence. These parameters include field-of-view (FOV), maximum gradient amplitude, maximum slew rate, and global radiofrequency (RF) frequency and phase shifts. The child nodes of the root node are called *loop nodes*. These control the loops or iterations in the sequence (e.g., phase encoding steps). The children of loop nodes are called *block nodes* and represent blocks of pulse sequence events. A block often corresponds to a single repetition in the pulse sequence, but can also encapsulate a preparation pulse or any portion of the repetition. Finally, children of block nodes are *segments*, or collections of low-level events. The basic segment types are excitation segments (consisting of a RF pulse event as well as pre-phasing and slice-select gradients), acquisition segments (consisting of a readout event as well as prephasing, encoding and readout gradients), refocusing segments (similar to excitation segments with an option for applying crusher gradients), and encoding segments (consisting of gradients for rewinding and/or spoiling). In addition other custom segment nodes may be defined by the user.

Foundation nodes versus custom nodes

Most ST nodes are foundation nodes, or built-in nodes that are building blocks for any pulse sequence. These include the segments listed above as well as basic blocks (gradient echo,

etc.) and loops (Cartesian sampling, etc.) Most sequences include variations on the basic techniques, and therefore the loops and blocks are customized by the programmer. However, since virtually all sequences are built out of collections of basic segments (excite, acquire, refocus, and encode), segments are usually not customized.

In some instances, a single event within a segment needs to be customized without modifying the rest of the segment, for example, a RF waveform within an excite segment. ST provides a way to handle this using class inheritance by allowing the user to *promote* a subnode without affecting the parent node.

Pulse sequence parameters

As described above, each node comprises a snippet of C++ code and a collection of parameter values that affect the functionality of the node. The code contains macros that define the parameter names, types, and default values as well as macros that define the child nodes (names and types). In addition, the code contains logic (executing at run-time) that may set parameters of child nodes and descendant nodes. As a general rule, nodes may not set parameter values for parent nodes -- the one exception being certain parameters of the root node which affect the sequence as a whole and may be modified at any time by any node.

Parameters that are set during pulse sequence execution (either by the node itself or by a parent or ancestor node) are called *automatic parameters*, whereas parameters that are not set programmatically are called *manual parameters*. Manual parameters may be set by the user without modifying the C++ code, whereas automatic parameters cannot be overridden, except through manipulation of the code. For example, in an excite segment, the slice thickness is a manual parameter while the RF bandwidth and RF duration are automatic parameters since they are controlled in the code according to the prescribed slice thickness and slice-select gradient amplitude.

Graphical user interface

Perhaps the most important component of ST is its graphical user interface (GUI). Figures 2 and 3 show the main windows of the GUI including the tree window, the parameter window, the code window, the sequence diagram window, and the event detail window. The user naturally interacts with individual nodes (in the sequence tree) using mouse clicks and easily modifies the parameters and C++ code for each node within the GUI. Each time changes are made to the sequence, the sequence diagram window is updated in real time. Users may also scroll through the individual event blocks in chronological order. Information about each event (e.g., for a RF pulse or individual gradient) is obtained by simply clicking on the event.

To maintain modularity in the sequence functionality, the logical interaction between nodes is minimized, allowing the user to modify selected portions of the sequence tree (such as changing an excitation pulse or switching from a Cartesian to radial loop) without affecting other parts of the tree. This also allows for copying and pasting of sequence objects. For

example, a custom RF pulse, a specialized imaging block, or even an entire loop from one sequence can easily be incorporated into an entirely different sequence via copy and paste.

Automatic encoding moments

To achieve a simpler and more intuitive programming experience, ST tracks the *k-space* location (or *k-location*) throughout the pulse sequence so that the programmer has the option of programming encoding gradients by specifying only the desired destination location in *k-space*. The *k-space* tracking is performed according to the following rules. Whenever an excitation pulse is encountered, the *k-location* is reset to the origin at the reference time of the pulse (the reference time is usually at the center or peak time of the pulse), and then moved through *k-space* according to the remaining slice-selective gradient (after the reference time). Similarly, when a refocusing pulse is encountered, the *k-location* is moved according to any gradients before the reference time, reflected across the origin at the reference time, and then moved again according to gradients after the reference time. Finally, when gradient pulses are encountered, the *k-location* is moved according to the total moment of the gradient pulses.

As an example, suppose that the programmer wants to program a fully balanced sequence such as balanced steady-state free precession (bSSFP). Normally, the programmer would need to compute the appropriate moment of the rewind gradient to return to the origin of *k-space* at the end of each repetition. However in ST, the programmer only specifies to return to location (0,0,0) using an encoding segment. The framework will then automatically insert a gradient with the appropriate duration, amplitude, and ramp times according to the maximum gradient strength and slew rates prescribed for the sequence. This works similarly for phase encoding gradients, where the programmer only needs to specify the desired location (0,k,0) in *k-space* in an acquire segment, and the framework will translate this location into gradient moment (according to the FOV) and then insert the appropriate gradients (including pre-phasing for the readout).

Automatic field-of-view shifts

Another important aspect of pulse sequence programming handled automatically by ST is the implementation of FOV shifts. Normally, the programmer would need to add appropriate phase and frequency offsets throughout the sequence to achieve scanning at off-isocenter locations. ST uses the technique outlined in (5) to automatically set these phase and frequency shifts, thus removing this burden from the programmer.

Custom RF pulses

Two approaches for using custom RF pulse waveforms are provided in ST: the functional approach, and the sampled approach. In the functional approach, the user supplies an overloaded (or replacement) member function that defines the real and imaginary parts of the RF waveform from an analytical formula. The code defining a simple Gaussian pulse is provided in an example sequence at the ST website. Once the waveform is defined, and manual parameters, e.g. bandwidth, flip angle, are set, ST automatically adjusts the RF pulse amplitude and duration accordingly. In some cases, analytical formulae may not be available for a RF pulse waveform, such as pulses designed by the Shinnar-Le Roux (SLR) algorithm.

The sampled approach can be used to import these discretized waveforms into ST. The user first designs a RF pulse waveform with a third-party pulse design tool or their own algorithm, then creates a text file containing the real and imaginary samples, along with its reference bandwidth and refocusing time. After uploading the text file to ST, the pulse samples are scaled in time and amplitude according to the prescribed bandwidth relative to the reference bandwidth and flip angle.

Non-trapezoidal gradient shapes

At the time of this publication, ST does not have the capability of handling arbitrary gradient shapes (e.g., spiral trajectories). This is not an inherent limitation, however, and we expect that this feature can be added without too much difficulty.

Exporting to a scanner

ST has the critical capability of exporting pulse sequences as programs that may be compiled and run on Siemens research MRI scanners. When the user selects this option, the framework creates a folder on the user's computer with a collection of files necessary for compilation. Since Siemens research sequences are written in C++, the code snippets from the ST nodes are simply inserted at the appropriate locations in the exported program. The main task of the export module is to translate ST events (which are independent of any particular scanner framework) into events that may be understood by the Siemens system.

Thus far only a Siemens interface module has been created, but the software has been designed to be scanner independent. As compared with the 56,000+ lines of scanner-independent code in the ST open source repository, the Siemens export module only consists of 1,520 lines. Ultimately, users should be able to export to any research scanner once the appropriate export modules are created.

Since ST is a research tool, sequence parameters are not, by default, adjustable at the scanner (as in clinical sequences), and are usually hard-coded during sequence development. One exception is the field-of-view positioning which may be set at the time of the scan. For convenience, ST has the option of specifying select custom parameters to be linked to so-called *special parameters* on the Siemens operator console so that they may be adjusted during scan time. In addition, standard parameters such as *echo time*, *repetition time*, and *flip angle*, may also be optionally linked to the corresponding standard parameters on the console.

Parsing of raw data

At the time of this publication, ST does not interface to the image reconstruction component of the scanner. Therefore, when sequences are run at the scanner, no image appears at the scanner console. However, the raw k-space data may be saved and images may be reconstructed offline. ST has a module that simplifies the step of parsing the raw data. The user opens the pulse sequence in ST, selects the option to read the raw data, and then selects the data file from the scanner. The framework parses the data file according to the order of readout events throughout the sequence, and organizes these data into separate raw data arrays, one for each readout segment.

Simulating MRI experiments

In addition to exporting to an actual scanner, ST can export sequences to a simulator (or virtual MRI scanner) so that pulse sequences may be thoroughly tested (using virtual phantoms) prior to an actual experiment. The built-in simulator of ST can run efficient simulations on large, 3-dimensional phantoms, and includes simulated noise and field inhomogeneity profiles. At each time point during the sequence, the magnetization of each tissue type is modeled as a sum of (possibly thousands of) transverse and longitudinal components. Each component is a pair consisting of a complex weight along with a point in 4-dimensional k-space (3 spatial frequency dimensions and 1 time dimension). An instantaneous RF pulse splits each term into 3 terms (1 longitudinal and 2 transverse), while applied gradients simply shift the terms in k-space. As terms evolve with T1- and T2-relaxation, they become negligible and are dropped according to a preset threshold. When a readout event is encountered, the transverse terms are all evaluated on the virtual k-space which is the preloaded Fourier transform of a large 3D spin-density array. The values for each tissue type are summed together to generate the raw data, as though it had come from an actual scanner. In this way, many artifacts are faithfully preserved for the purpose of sequence debugging.

Experiments

ST was used by dozens of researchers, primarily in the authors' laboratory over the course of eight years. Hundreds of pulse sequences were created and run on Siemens research scanners (Sonata, Avanto, and Tim Trio, Magnetom) at 1.5, 3, and 7 T field strengths. Although sequences were not tested on some of the most recent Siemens scanners (Skyra, Prisma, Verio, and Aera), the versions of the scanner software for those scanners match those of the newest Avanto scanners (VD13) at our institution, so no major compatibility issues are expected with the above systems, at least at the present time.

RESULTS

The software has been used in more than 50 peer-reviewed journal articles in diverse areas of NMR Research. The applications include dynamic measurement of blood oxygenation and blood velocity (7,8), measurement of pulse-wave velocity (9), simultaneous T2* and perfusion quantification in the leg (10), vessel wall imaging (11), high-resolution imaging of trabecular bone (12,13), sodium imaging at the knee (14), MR elastography (15), spectrally selective 3D fast spin echo (16), dynamic imaging of phosphocreatine recovery kinetics (17), and solid-state ^1H and ^{31}P cortical bone imaging (18,19). Despite the commonalities between many of the sequences for these applications, each of the above techniques required individual customization.

The goal of ST is to enable users to make modifications to pulse sequences with minimal manipulation of source code. Table 1 shows the numbers of lines of code and parameter changes required to add basic features to a ST pulse sequence. The actual code for these examples can be found at the ST website (<http://www.sequencetree.org>).

An example of ST being used to implement a novel pulse sequence is provided in a recent study involving dynamic measurement of CMRO₂ (cerebral metabolic rate of oxygen) in humans during rest and induced hypercapnia (7,20). The sequence involves interleaving of two simpler sequences -- one for quantifying S_vO₂ (venous oxygen saturation) at the level of the superior sagittal sinus, and the other for quantifying blood flow at the level of the internal carotid arteries. The acquisition alternates between the two slices (see Figure 4) in an interleaved manner, but the pulse sequence is different at the two locations. The simultaneous acquisition of these two quantities enables dynamic estimation of CMRO₂. This sequence exemplifies the utility of ST for designing and implementing new, yet simple, acquisition strategies that are unlikely to have been part of the *product sequences* of the scanner vendors.

A second example of a new pulse sequence implemented in ST is 3D FSE-OSC (fast spin echo with out-of-slab cancellation) (13). This variant of fast spin echo uses non-selective refocusing pulses to reduce the specific absorption rate (SAR), which is especially important at ultra-high field strengths. These non-selective pulses excite a substantial amount of magnetization outside the imaging slab. To remedy this problem, the phase of the excitation pulse is alternated between 0° and 180° and the two images are subtracted to cancel spurious signal from outside the imaging slab. Another custom feature incorporated into the 3D FSE-OSC sequence (and easily implemented in ST) is a variable (decreasing) flip angle for successive pulses in the echo train. Images obtained by applying the 3D FSE-OSC sequences *in vivo* at the distal tibia are shown in Figure 5.

A third example of data acquired using ST is a projection based method for obtaining a time-resolved velocity measurement of blood in the femoral artery (8). After acquiring a 2D reference image, a 1-dimensional readout is repeated with alternating velocity-encoded gradients. A phase difference is computed for each pair of velocity-encoded projections after removing the stationary tissue signal (estimated from the reference image). This allows time-resolved absolute quantification of blood flow velocity without gating. Because this novel sequence combines a 2D acquisition with a series of repeated 1D projections, it cannot be programmed as a simple variant of a *built-in* pulse sequence provided by the scanner vendor, and is therefore ideally suited for a ST implementation (see Figure 6).

An illustration of how the virtual scanner of ST can be used to detect and diagnose artifacts is shown in Figure 7. A researcher in the authors' laboratory made a subtle error while programming a novel pulse sequence by leaving out a spoiler gradient resulting in a stimulated echo artifact. The sequence was a complex variant of spoiled gradient echo where multi-band excitation was used to suppress inflowing blood signal. Without the spoiler gradient, signal from one slice was leaking into the imaging slice in a later repetition.

DISCUSSION

The key advantage of ST is that it seamlessly combines a visual environment with a coding environment for intuitive and flexible pulse sequence programming. The practical utility of the software is reflected in the large number of new pulse sequences that have been designed, implemented and tested, predominantly by researchers with little or no computer

programming experience. Many of these sequences are currently being used in ongoing clinical research studies at the authors' institution and elsewhere.

This integration of WYSIWYG with traditional code-based programming is made possible by various automations built into ST. One such feature is automatic encoding gradients where the programmer specifies the desired k-space location at a certain time during the sequence, and the framework will automatically achieve this by setting the optimal gradient amplitudes and ramp times. Another feature allows for automated phase and frequency adjustments to achieve any off-center field-of-view position, independent of the pulse sequence.

ST is not suited for all use cases. As mentioned in the methods section, ST has not yet been interfaced to the image reconstruction component of the Siemens scanner, so no image appears on the console at scan time. Obviously this limits the scope of ST sequences to research applications where data is reconstructed and analyzed offline. Another limitation is that ST sequences are always built from scratch and are not capable of using sequence modules (or building blocks) provided by the scanner manufacturer. If the user wants to make small adjustments to a pulse sequence provided as a product by the manufacturer, then it is usually preferable to work with the source code (if available) of the base sequence. On the other hand, the ST environment is most appropriate when the programmer desires to maintain full control over all aspects of the pulse sequence without worrying about unnecessary complexities of the broader programming environment. In general, depending on the sequence and application, there may be advantages of using ST for prototyping proof-of-concept techniques, but in other cases it may be preferable to modify vendor source code directly.

Since the core components of ST are independent of any particular type of scanner, sequences can, in principle, be exported to any MRI scanner, provided that a suitable export module is created. At the time of this publication, ST sequences may only be exported to Siemens scanners. This is admittedly a limitation of the software, especially for researchers without access to these systems. It is the authors' hope that this publication will spark interest among researchers with access to non-Siemens scanners, perhaps to assist in creating additional export modules. More than 97% of the source code of ST is scanner-independent, and therefore providing additional interfaces would likely not require a significant restructuring of the core program.

There are, however, a few obstacles that could potentially preclude a straightforward interface to the other scanners. First and foremost, the underlying programming language of the Siemens programming environment (as well as for ST) is C/C++. This obviously poses a serious hurdle for scanners with environments built on other programming languages. However, this challenge is not necessarily insurmountable. One *brute-force* remedy would be to have ST output a binary file containing a list of all sequence events, and then have a wrapper program read the file during sequence execution, running the events on the fly. Another possibility is to rewrite portions of ST such that the low-level event implementations are specific to the target scanner, and then the higher-level code and logic would need only minimal modifications. Even after this hurdle is overcome, there are other

challenges related to interfacing with the operator interface of the scanner console. Tackling these hurdles would require a concerted collaborative effort among researchers with experience in programming the target scanner types.

In addition to the technical aspects of pulse sequence programming, there are many legal and practical issues associated with sharing pulse sequences between institutions and across scanners and software versions. In the interest of reproducible science, these issues will need to be addressed. Such a discussion is considered beyond the scope of the present article.

One concern sometimes raised by users of ST is whether the framework bypasses the safety checks for power deposition and nerve stimulation. In the case of Siemens scanners, these parameters are checked by the software of the scanner vendor (not by ST) and so no safety checks are bypassed. As modules are created for other scanner types, this issue will need to be addressed on a case-by-case basis. A limitation of ST is that the safety checks are only executed at scanner run time, not during the sequence design phase. However, as in conventional Siemens sequences, the framework will provide the user with the option of increasing the repetition time, as long as this parameter has been interfaced during export. Similarly, parameters such as maximum gradient amplitude and gradient slew rate may also be adjusted in this manner provided that, again, these parameters are interfaced during export. The interfacing of these options requires effort on the part of the sequence programmer and is not implemented by default.

An advantage of programming with ST is that sequences may be simulated using the virtual scanner of ST prior to running an actual experiment. Both pulse sequence programming errors as well as subtle design issues may be detected on the basis of sequences applied to virtual digital phantoms. Since the very same pulse sequences are applied during simulation and at the actual scanner, problems may be diagnosed and corrected ahead of time, thus preventing a waste of scanner time and resources in many cases.

The ST user interface provides a pulse sequence diagram that updates immediately with every parameter or code change. Therefore it is ideally suited for educational and training purposes. Users who are new to NMR and/or pulse sequence design may adjust basic sequence parameters (echo time, repetition time, readout/excitation bandwidth, flip angle, etc.) and visualize the results in the pulse diagram pane. For more advanced users, adjustments to the C++ code controlling the hierarchical structure of the sequence and all other aspects of the program may be debugged and tested, all within the same user interface.

CONCLUSION

ST is an open source, visual pulse sequence environment for MRI that combines simplicity with flexibility. Pulse sequences may be tested using a virtual scanner and exported for use with an actual scanner. The core design of ST is scanner-independent, and it is anticipated that export modules will ultimately be created for non-Siemens scanner platforms. The software has already been used in numerous peer-reviewed publications featuring novel acquisition techniques and is ideal for researchers with limited computer programming experience.

Acknowledgments

Grant Sponsor: National Institutes of Health; **Grant number:** K25 AR 060283

The authors thank the users of ST for their feedback. This work was supported by National Institutes of Health grant K25 AR 060283.

References

1. Jochimsen TH, von Mengershausen M. ODIN-object-oriented development interface for NMR. J Magn Reson San Diego Calif 1997. 2004; 170:67–78.10.1016/j.jmr.2004.05.021
2. Overall, WR.; Pauly, JM. An extensible, graphical environment for pulse sequence design and simulation. Proceedings of ISMRM, 15th Annual Meeting; Berlin, Germany. 2007; p. 1652
3. Stöcker T, Vahedipour K, Pflugfelder D, Shah NJ. High-performance computing MRI simulations. Magn Reson Med Off J Soc Magn Reson Med Soc Magn Reson Med. 2010; 64:186–193.10.1002/mrm.22406
4. Hansen MS, Sørensen TS. Gadgetron: an open source framework for medical image reconstruction. Magn Reson Med. 2013; 69:1768–1776.10.1002/mrm.24389 [PubMed: 22791598]
5. Magland J, Wehrli FW. General algorithm for automated off-center MRI. Magn Reson Med Off J Soc Magn Reson Med Soc Magn Reson Med. 2006; 56:230–233.10.1002/mrm.20933
6. Magland, JF.; Wehrli, FW. Pulse sequence programming in a dynamic visual environment. Proceedings of ISMRM, 14th Annual Meeting; Seattle, Washington. 2006; p. 3032
7. Jain V, Langham MC, Floyd TF, Jain G, Magland JF, Wehrli FW. Rapid magnetic resonance measurement of global cerebral metabolic rate of oxygen consumption in humans during rest and hypercapnia. J Cereb Blood Flow Metab Off J Int Soc Cereb Blood Flow Metab. 2011; 31:1504–1512.10.1038/jcbfm.2011.34
8. Langham MC, Floyd TF, Mohler ER, Magland JF, Wehrli FW. Evaluation of cuff-induced ischemia in the lower extremity by magnetic resonance oximetry. J Am Coll Cardiol. 2010; 55:598–606.10.1016/j.jacc.2009.08.068 [PubMed: 20152564]
9. Langham MC, Li C, Magland JF, Wehrli FW. Nontriggered MRI quantification of aortic pulse-wave velocity. Magn Reson Med Off J Soc Magn Reson Med Soc Magn Reson Med. 2011; 65:750–755.10.1002/mrm.22651
10. Englund EK, Langham MC, Li C, Rodgers ZB, Floyd TF, Mohler ER, Wehrli FW. Combined measurement of perfusion, venous oxygen saturation, and skeletal muscle T2* during reactive hyperemia in the leg. J Cardiovasc Magn Reson Off J Soc Cardiovasc Magn Reson. 2013; 15:70.10.1186/1532-429X-15-70
11. Langham MC, Li C, Englund EK, Chirico EN, Mohler ER, Floyd TF, Wehrli FW. Vessel-wall imaging and quantification of flow-mediated dilation using water-selective 3D SSFP-echo. J Cardiovasc Magn Reson Off J Soc Cardiovasc Magn Reson. 2013; 15:100.10.1186/1532-429X-15-100
12. Magland JF, Wald MJ, Wehrli FW. Spin-echo micro-MRI of trabecular bone using improved 3D fast large-angle spin-echo (FLASE). Magn Reson Med Off J Soc Magn Reson Med Soc Magn Reson Med. 2009; 61:1114–1121.10.1002/mrm.21905
13. Magland JF, Rajapakse CS, Wright AC, Acciavatti R, Wehrli FW. 3D fast spin echo with out-of-slab cancellation: a technique for high-resolution structural imaging of trabecular bone at 7 Tesla. Magn Reson Med Off J Soc Magn Reson Med Soc Magn Reson Med. 2010; 63:719–727.10.1002/mrm.22213
14. Madelin G, Lee J-S, Inati S, Jerschow A, Regatte RR. Sodium inversion recovery MRI of the knee joint in vivo at 7T. J Magn Reson San Diego Calif 1997. 2010; 207:42–52.10.1016/j.jmr.2010.08.003
15. Cortes DH, Magland JF, Wright AC, Elliott DM. The shear modulus of the nucleus pulposus measured using magnetic resonance elastography: a potential biomarker for intervertebral disc degeneration. Magn Reson Med Off J Soc Magn Reson Med Soc Magn Reson Med. 2014; 72:211–219.10.1002/mrm.24895

16. Parasoglou P, Xia D, Regatte RR. Spectrally selective 3D TSE imaging of phosphocreatine in the human calf muscle at 3 T. *Magn Reson Med Off J Soc Magn Reson Med Soc Magn Reson Med*. 2013; 69:812–817.10.1002/mrm.24288
17. Parasoglou P, Xia D, Chang G, Regatte RR. Dynamic three-dimensional imaging of phosphocreatine recovery kinetics in the human lower leg muscles at 3T and 7T: a preliminary study. *NMR Biomed*. 2013; 26:348–356.10.1002/nbm.2866 [PubMed: 23065754]
18. Li C, Seifert AC, Rad HS, Bhagat YA, Rajapakse CS, Sun W, Lam SCB, Wehrli FW. Cortical Bone Water Concentration: Dependence of MR Imaging Measures on Age and Pore Volume Fraction. *Radiology*. 2014; 272:796–806.10.1148/radiol.14132585 [PubMed: 24814179]
19. Seifert AC, Li C, Rajapakse CS, Bashoor-Zadeh M, Bhagat YA, Wright AC, Zemel BS, Zavalangos A, Wehrli FW. Bone mineral (31)P and matrix-bound water densities measured by solid-state (31)P and (1)H MRI. *NMR Biomed*. 2014; 27:739–748.10.1002/nbm.3107 [PubMed: 24846186]
20. Wehrli FW, Rodgers ZB, Jain V, Langham MC, Li C, Licht DJ, Magland J. Time-resolved MRI oximetry for quantifying CMRO(2) and vascular reactivity. *Acad Radiol*. 2014; 21:207–214.10.1016/j.acra.2013.11.001 [PubMed: 24439334]

Figure 1.

Hierarchical sequence structure for a basic spoiled gradient echo sequence. Each node shows an object name and the name of the corresponding C++ class (in parenthesis). Green nodes are 'loop' objects, red nodes are 'block' or 'segment' objects, and blue nodes represent low-level scanner events.

Figure 2. Screenshot of the ST user interface showing the tree window (left), parameter window (top right), and pulse diagram window (bottom right). As sequence parameters are modified using mouse clicks, the pulse diagram is updated in real time. This sample sequence features a dual gradient echo. Readout, phase encode, and slice gradients are represented as GX, GY, and GZ, respectively.

Figure 3.

Screenshot of the ST user interface showing source code for the sequence modifications required to achieve an additional echo. The resulting sequence is illustrated in Fig. 1. As shown in Table 1, five lines of code (highlighted by arrows) are needed to achieve this modification.

Figure 4.

Images generated using a ST pulse sequence. The acquisition alternates between two interleaved sequences at different locations. The first sequence measures oxygen saturation at the level of the sagittal sinus while the second sequence simultaneously measures blood flow at the level of the internal carotid arteries. *From Jain et al. (7), with permission.*

Figure 5. High-resolution image of the distal tibia acquired using 3D FSE-OSC implemented using ST. The sequence features variable flip angle, non-selective refocusing pulses and phase cycling, as well as subtraction of two interleaved acquisitions to obtain out-of-slab signal cancellation. *From Magland et al. (13), with permission.*

Figure 6.

Results obtained from a time-resolved velocity measurement implemented using ST. (a) Representative reference image of the femoral artery at the level of the mid-thigh. (b) Time series of 224 velocity projection images. Dashed lines encompass region of velocity averaging along the readout direction. (c) Time course of the average velocity in the femoral artery at 20 ms temporal resolution. *From Langham et al. (8), with permission.*

Figure 7.
Comparison of phantom images from an actual scanner (A and C) with the virtual scanner of ST (B and D) with and without an inadvertent pulse sequence programming error.

Table 1

Number of programming changes required for various basic pulse sequence modifications.

Action	No. code lines added	No. parameters changed
Add fat saturation pulse	6	4
Add RF spoiling	2	0
Cartesian to radial	8	0
Add additional echo	5	0
Change polarity of the echo	1	0
Change RF waveform	8	0
Change TE, TR, flip angle, field-of-view, spoil gradient moment, etc.	0	1