HHS Public Access

Author manuscript

Int J Comput Assist Radiol Surg. Author manuscript; available in PMC 2018 March 01.

Published in final edited form as:

Int J Comput Assist Radiol Surg. 2017 March ; 12(3): 449–457. doi:10.1007/s11548-016-1495-z.

GRAPE: A graphical pipeline environment for image analysis in adaptive magnetic resonance imaging

Refaat E. Gabr[1], **Getaneh B. Tefera**[1], **William J. Allen**[2], **Amol Pednekar**[3], and **Ponnada A. Narayana**[1]

[1]Departments of Diagnostic and Interventional Imaging, University of Texas Health Science Center at Houston (UTHealth), Houston, TX

[2]Texas Advanced Computing Center, University of Texas at Austin, Austin, TX

[3]Philips Healthcare, Cleveland, OH

Abstract

Purpose—We present a platform, GRAphical Pipeline Environment (GRAPE), to facilitate the development of patient-adaptive magnetic resonance imaging (MRI) protocols.

Methods—GRAPE is an open source project implemented in the Qt C++ framework to enable graphical creation, execution, and debugging of real-time image analysis algorithms integrated with the MRI scanner. The platform provides the tools and infrastructure to design new algorithms, build and execute an array of image analysis routines, and a mechanism to include existing analysis libraries, all within a graphical environment. The application of GRAPE is demonstrated in multiple MRI applications, and the software is described in detail for both the user and developer.

Results—GRAPE was successfully used to implement and execute three applications in MRI of the brain, performed on a 3.0 Tesla MRI scanner: (i) a multi-parametric pipeline for segmenting the brain tissue and detecting lesions in multiple sclerosis (MS), (ii) patient-specific optimization of the 3D fluid-attenuated inversion recovery (FLAIR) MRI scan parameters to enhance the contrast of brain lesions in MS, and (iii) an algebraic image method for combining two MR images for improved lesion contrast.

Conclusions—GRAPE allows graphical development and execution of image analysis algorithms for inline, real-time, and adaptive MRI applications.

Keywords

graphical user interface; patient-specific imaging; advanced computing; real-time; visual programming

Correspondence to: Refaat E. Gabr (refaat.e.gabr@uth.tmc.edu), Department of Diagnostic and Interventional Imaging, University of Texas Medical School at Houston, 6431 Fannin St, MSE R102D, Houston, TX 77030, Tel: 713-500-7660, Fax: 713-500-7684.

Conflict of interest. The authors declare no conflict of interest.

Ethical standard. This research was approved by the Committee for the Protection of Human Subjects of the University of Texas Health Science Center at Houston.

Informed consent. Informed consent was obtained from all participants included in the study.

Introduction

Medical image analysis is an essential component of clinical research and healthcare decision making. It provides a wealth of information that characterizes and quantitatively assesses the disease state [1]. Notable advances in medical imaging and image analysis over the years have resulted in the development of several important analysis and visualization packages, algorithms, and libraries [2–7]. To support the increased use of these tools, frameworks that provide a graphical user interface (GUI) for the design of analysis pipelines have also emerged [8–12]. However, these frameworks are primarily designed for off-line data analysis; to date, integration with the magnetic resonance (MR) image acquisition system has not been considered as a factor in these frameworks. Thus, there is a need for a programming environment to enable the creation of image analysis pipelines that can be integrated with the magnetic resonance imaging (MRI) scanner and support an adaptive scan session. Real-time response will result in improved image contrast, immediate quantitative data, and same-session image quality control for minimizing the need for patient call-backs. The main requirements for such an environment are simplicity, computational efficiency, design flexibility, and extensibility.

Here, we present an extensible GRAphical Pipeline Environment (GRAPE) for the design and execution of a MR image analysis pipeline in an adaptive scan environment. GRAPE shares many similarities with other visual programming tools. However, GRAPE is designed with the goal to facilitate real-time feedback, synchronize image analysis with data acquisition, and provide a user-friendly framework for adaptive protocols. Therefore, GRAPE is built with flexibility to communicate data with the scanner, and perform MRI pulse sequence optimization and other non-image based calculations. GRAPE also provides a simple and flexible cross-platform algorithm development environment that allows users to generate new algorithms, analyses, and pipelines. In this manuscript we describe the software architecture in GRAPE and a selection of the programming tools available to the user and the developer. We also present three case studies, as examples, using GRAPE to design and run image analysis pipelines in MRI, including adaptive MRI scans where the results of the computations are used to adjust in pseudoreal time the scan parameters specific to the individual patient [13].

Methods

GRAPE Layout

The GRAPE graphical interface consists of the pipeline layout panel and the pipeline development toolbar (Fig. 1). The pipeline layout panel is an interactive canvas that allows the user to create and edit various modules of the analysis pipeline (as nodes), and control the data flow (as edges). The pipeline development toolbar includes basic drawing tools, execution buttons, and the module library. The module library shows all the available modules for the user, including both GRAPE built-in and user-created libraries. From the toolbar, the user can also start and stop the execution of the current pipeline.

An image analysis pipeline is built by selecting the desired modules from the available libraries and adding it to the canvas. For example, there are modules for reading data,

performing image processing tasks, and writing output. The nodes are connected according to the desired data flow plan of the pipeline. For each module, a property dialog is defined and can be accessed from the module context menu. Through the module property dialog, various attributes such as the legend, rendering options, and functional parameters can be modified.

Selecting the *Play* button puts the pipeline in the run mode. Once the pipeline is in the run mode, editing of the modules is disabled. During execution, the modules show color cues indicating the current progress of the pipeline, and time-stamped events are automatically written to a log file. The user can also include additional logging statements. Another feedback mechanism is available with image display modules programmed to display 2D, 3D, or 4D images, allowing the user to review the pipeline outputs at various stages. This is particularly useful during the initial development stages of analysis algorithms. The pipeline periodically checks the status of each node, and re-executes the nodes when their input data are changed. This is useful for nodes that require, for example, input from two separate branches in the pipeline. The user may interrupt the execution of the pipeline at any time by pressing the *Stop* button.

GRAPE stores pipelines as extended mark-up language (XML[14]) scripts with all settings, parameters, and options embedded. These XML scripts can be executed on the command line or in the GRAPE GUI. The user may edit the file to directly change the parameters of the modules as well as through the GRAPE graphical environment. The pipeline files may also be saved, shared, and restored between users.

Pipeline Tools

GRAPE includes basic programming elements such as logical operations, list selection, branching, and looping. These are all implemented as modules that the user can modify and use directly without the need for programming, allowing quick prototyping of new pipelines. In addition, these tools are also useful for pipelines that perform batch processing of multiple datasets. Special *Source* and *Sink* nodes provide convenient means to generate test data and parameters, display analysis outputs, and manually debug pipelines.

Basic image operations are implemented to facilitate the development of new algorithms. The image library includes modules for image arithmetic, slicing, statistics, and comparison. Widely-used tools such as BET brain extraction [15], nonuniformity correction methods [16], image registration [17], and segmentation [18] are also available. Reading in and writing to popular image formats such as DICOM, Analyze and NIFTI is supported. Since the pipeline is implemented on Philips MRI scanner, read/write is supported for the Philips XML/REC research format. The latter is crucial for allowing data exchange directly between the Philips MRI scanner and the analysis pipelines.

External user programs and image analysis functionalities in third party applications are made available through command line calls to executable files. The GRAPE interface allows the user to create a customized system-call module to define the inputs and outputs to and from the executable program. This provides a high degree of flexibility for the user, avoids

the need for re-programming tasks, and allows the extension of the program with a moderate effort.

Scanner Integration

The GRAPE environment has been seamlessly integrated with an MRI scanner. A vendor-specific software tool interfaces to the scanner database and performs automated extraction of the images immediately after the completion of data acquisition and image reconstruction. The vendor tool also initiates the transfer of data to the post-processing pipeline, and remotely launches GRAPE with the desired processing pipeline as an argument. GRAPE can be launched in both design and run modes, with the graphical display as an option. The *Source* node in GRAPE can be assigned to watch for input files for the pipeline. After the pipeline finishes execution (an *End* node is encountered in GRAPE), or when specific result files are written, the pipeline outputs are transferred back to the scanner, including data, images, and scan parameter files. The images are then automatically imported back into the patient database on the scanner, and are available for review by radiologist either on the scanner or via transfer to PACS. The whole process of data export, transfer, processing, and importing the results is fully automated. In principle, any vendor-specific software that can interface to the scanner database and execute external batch scripts can easily integrate with GRAPE, although this has not been tested. Unlike PACS workstations or PACS-like research platforms like XNAT [19], our framework integrates GRAPE with the scanner as early as the time of the data acquisition.

Module Development

GRAPE is implemented in the Qt 5.4.2 C++ application development framework. It uses elements from the open-source projects JADE (https://sourceforge.net/projects/jade-diagram) used for diagram editing, and from the Template Image Processing Library (https://www.nitrc.org/projects/tipl/) for handling DICOM, NIFTI, and Analyze image file formats. GRAPE uses the Qt build tool, QMake, and was successfully compiled for Windows, Linux, and Mac operating systems without modifications, and was run on both standalone computers as well as on the *Stampede* Dell PowerEdge Linux cluster at the Texas Advanced Computing Center (https://www.tacc.utexas.edu/stampede, Austin, TX, USA).

Users can implement new modules by defining a *Node Interface* descriptor in the node definition file. The descriptor specifies the number of inputs and outputs, the type and default values of the inputs and outputs, and which inputs are mandatory. The *Node Interface* is used to populate the library viewer in the toolbar and to dynamically generate the graphical representation of the node. The functional component of the module is implemented by a *Node* class. Each node implementation must define four basic methods: input assignment, output assignment, validation, and execution. A special *Develop* node is implemented in GRAPE for quick prototyping of new modules. The interface of this node is already implemented, and the user can directly insert the functional part of the algorithm.

The GRAPE source code is organized into three subdirectories, presenting the nodes, the graphical items, and core diagram handling. Separate subdirectories are added for other libraries. The module libraries are arranged in a tree structure under core GRAPE and user

modules. The core GRAPE modules are further subdivided into input/output, pipeline, and image operations. GRAPE currently includes 27 individual modules. Some of the modules (e.g. *ImageArithmetic*) handles a class of image operations, all grouped into a single interface allowing the user to select from the individual functions. New modules and libraries are currently under development and will be part of the first GRAPE release.

For the case reports to demonstrate the application of GRAPE, the MRI data were acquired on a Philips 3.0 Tesla MRI scanner. Analyses were performed on a local Windows computer with 3.2-GHz 8-core Intel Core i7 and 12 GB RAM.

Results

To demonstrate the use of GRAPE modules, Fig. 2 shows a pipeline used to test the effect of the value of the intensity threshold on BET brain extraction when applied to fat-saturated T2-weighted MR images. In this case, the user can visually experiment with the parameter(s) of interest and immediately see the results without the need for coding. The pipeline XML file of this pipeline is available as supplementary material.

We implemented three separate image analysis pipelines aimed at improving image contrast and quantitative MRI. The first application is a multi-parametric brain tissue pipeline for tissue segmentation and detecting brain lesions in multiple sclerosis (MS). The second application is a pipeline for patient-specific optimization of 3D fluid-attenuated inversion recovery (FLAIR) sequence. This pipeline optimizes the scan parameters during a single scan session in order to enhance the contrast of brain lesions in MS. The third application is implementation of an algebraic image method for combining two MR images to generate an image with improved lesion contrast.

Brain Tissue Segmentation in MS

MS is an autoimmune, inflammatory, and demyelinating disease of the central nervous system that affects 2–2.5 million worldwide [20]. MRI plays a key role in the diagnosis and management of MS [21]. A hallmark of MS is the presence of white matter (WM) lesions, which typically show up on MRI as hyper-intense regions on T2-weighted (T2w) and FLAIR images. Several automated techniques have been proposed to segment the WM lesions from MRI images (e.g. [22–25]). We used GRAPE to implement an advanced algorithm for the automated analysis of multi-channel MRI data [25].

The imaging protocol in this study included the acquisition of dual-echo fast spin-echo and FLAIR. The dual-echo data provided T2w and proton density (PD) weighted images. As described below, the preprocessing pipeline incorporated published standard methods. The analysis included the following steps. First, brain extraction was performed to remove extra-meningeal tissues [15]. Second, all images were corrected for intensity non-uniformity [16]. Next, co-registration of FLAIR and T2w images was performed using affine transformation [3], orienting the three images in the same space. Finally, multi-parametric segmentation using the PD, T2w, and FLAIR images was performed to classify the brain into WM, grey matter (GM), cerebrospinal fluid (CSF), and WM lesions. The segmentation used in this

work employed a hybrid parametric and nonparametric approach [25]. The pipeline to execute this analysis was assembled in GRAPE and is shown in Fig. 3.

This pipeline demonstrates the capability of incorporating libraries from other sources through the command line interface module. In addition, integration of the pipeline into the scan environment allowed the implementation of acquisition-synchronized image analysis. In other words, instead of postponing processing until all data has been acquired, the scan integration allowed processing part of the data while the MRI was actively acquiring other data. This approach was demonstrated here by processing the dual-echo images (Fig. 3, left branch) while the scanner was still acquiring the FLAIR data (Fig. 3, right branch). As soon as the FLAIR acquisition was completed, the FLAIR images were automatically imported in the pipeline, co-registered with T2w, and processed. Finally, brain tissue segmentation was performed to generate the tissue maps, including WM lesion, and the maps were automatically imported back into the MRI scanner, and subsequently added to the patient database. This technique reduced the waiting time by almost 50% (from 119 sec to 61 sec) and removed the necessity for human intervention to initiate the analysis. Fig. 4 shows representative images from the segmentation pipeline. A combined image showing all maps is color-coded for better visualization. In this example, the real-time segmentation data quality is identical to what would typically be computed off-line after the patient has left the scanner. Thus, the real time processing did not appear to compromise the quality of this result.

Patient-specific FLAIR Optimization

Recent work shows that patient-specific optimization of the scan parameter may improve the image contrast. This was shown for both double inversion recovery [13] in healthy subjects, and for FLAIR in patients with MS [26]. In patient-specific FLAIR, the parameters inversion time (TI) and echo time (TE) were optimized for each individual patient based on fast mapping of PD and the T1 and T2 relaxation times of the brain tissues. The graphical pipeline that demonstrates the data flow in FLAIR optimization is show in Fig. 5. Notably, this particular pipeline was originally implemented as a command-line script. The executable files used to perform the PD/T2 mapping, T1 mapping, and the optimization of the scan parameters of FLAIR were included in the graphical pipeline using the command line module, avoiding the need for reprogramming.

Similar to the brain segmentation pipeline, by processing the PD/T2 data during the acquisition of the T1-mapping pulse sequence, a reduction in the perceived processing time by approximately 46 sec (25%) was achieved. The optimized scan parameters were applied to acquire FLAIR, and the improved lesion-brain contrast resulting from the patient-specific tuning of the sequence parameters is shown in Fig. 6.

Algebraic Image Combination

Due to physiological constraints, no single MRI pulse sequence can generate all the desired tissue contrast characteristics. However, it is possible that post-acquisition combination of images acquired with different pulse sequences can produce images with improved contrast or accentuated features compared to the images from the individual pulse sequences. For

example, combination of 3D T2w and 3D FLAIR images produces new images with enhanced lesion-brain contrast and low signal from the cerebrospinal fluid [27, 28]

A pipeline to perform this technique and representative images are shown in Fig. 7. This pipeline constructed the $FLAIR_3$ image from the T2w and FLAIR images using the formula [28]: $FLAIR_3 = (FLAIR^{1.55}) \times (T2w^{1.45})$. Image arithmetic was performed with built-in GRAPE modules. This pipeline also enables the user to experiment with different parameters of the image combination, while immediately visualizing the results.

Discussion

MRI produces high quality 2D and 3D images with a wide range of user-selectable contrast mechanisms. This flexibility makes MRI a powerful tool in research as well as in clinical applications. With careful selection of the scan parameters, superior image contrast can be realized to reflect subtle differences in tissue properties, such as proton density, relaxation times, perfusion state, metabolite levels and diffusion. Selection of the scan parameters is typically based on empirical approaches, or on the expected values of the tissue properties [29, 30]. However, the MRI protocol for a given patient population is usually fixed, and does not change from patient to patient. This approach is suboptimal at an individual patient level, and may result in a loss in sensitivity that would not otherwise occur if the protocol was optimized specifically for the patient.

Recent developments have shown the feasibility of optimizing the selection of scan parameters for the pulse sequence for each patient [13, 26]. Optimized scan parameters enhance image contrast and can contribute positively to the early identification of lesions and assessment of the state of disease. We anticipate that new scan protocols will emerge to take advantage of the ability to perform on-the-fly adjustments to the MRI pulse sequence parameters to better match the individual patient. The development of algorithms to perform such adjustments is facilitated by the availability of several image analysis packages. However, these packages are almost exclusively intended for off-line data analysis. Moreover, no single package is sufficient to handle all possible protocols. GRAPE addresses these problems by providing an environment that facilitates interfacing with the MRI scanner, integration of the available image analysis packages, and a flexible graphical programming environment.

Although scripts can offer similar pipeline implementation, and sometimes provide a high degree of flexibility and efficiency, visual programming offers many advantages. The ease of use and the speed of developing new algorithms are greatly enhanced, especially with the availability of a rich library of analysis modules. This is particularly beneficial for users with clinical insight (e.g. radiologists and physicians) to test new concepts. Moreover, the visualization afforded by the pipeline greatly facilitates the inspection of outputs, detection of execution bottlenecks, and the debugging of execution problems.

While the supported image operations built into GRAPE are still limited, GRAPE is an open source project and the code base will continue to grow with user contributions. The built-in command line interface allows the extension of GRAPE by incorporating functionalities

from other libraries. GRAPE is implemented in Qt C++, which requires learning for users/developers who are not familiar with Qt. However, the Qt C++ framework provides a platform-independent implementation, and is more efficient compared to virtual machine-based platforms such as JAVA. Users can also prototype new pipelines from existing modules or template pipelines with no programming experience.

GRAPE is still under active development, and new tools are being continuously added to improve the user experience. These include the powerful ITK [6] library and Slicer command line interface (CLI) modules [5]. We will also investigate the potential of GRAPE to run as a server application on high-performance hardware, such as the Stampede Linux cluster, and we will pursue further integration of GRAPE into the design process of MRI protocols. With the proof-of-concept implementation of adaptive MRI on Philips MRI systems, we anticipate its extension to other MRI manufacturers will follow. Propriety application software from the scanner manufacturers will be needed to allow proper scanner integration.

Rigorous testing is needed to assure robust performance of GRAPE under different pipelines. Unit testing of all modules has been manually performed using custom-built pipelines. In addition, the cases presented in this work provide additional testing. GRAPE is anticipated to be publically available in 2017 as an open source project, and will be released under the GNU General Public License version 3.0.

In summary, we present GRAPE, a graphical pipeline programming environment that allows the user to develop and execute image analysis algorithms for inline, real-time, adaptive MR imaging applications. Detailed user and developer documentation will be available to facilitate the installation and use of GRAPE, and to advance applications of adaptive scanning by the imaging community.

Supplementary Material

Refer to Web version on PubMed Central for supplementary material.

Acknowledgments

This work was supported by the Clinical Translational Science Award (CTSA) Grant UL1 TR000371 from the e National Institutes of Health (NIH) National Center for Advancing Translational Sciences, and the Chair in Biomedical Engineering Endowment Funds. Stampede was generously funded by the National Science Foundation (NSF) through award ACI-1134872. We thank Xiaojun Sun for useful discussion and Vipulkumar Patel for help with the MRI experiments.

References

1. Dhawan AP. Medical Image Analysis. Eng Med Biol. 2011:i–xv.
2. Cox RW. AFNI: software for analysis and visualization of functional magnetic resonance neuroimages. Comput Biomed Res. 1996; 29:162–173. [PubMed: 8812068]
3. Avants BB, Tustison NJ, Song G, Cook PA, Klein A, Gee JC. A reproducible evaluation of ANTs similarity metric performance in brain image registration. Neuroimage. 2011; 54:2033–2044. [PubMed: 20851191]
4. Friston KJ, Holmes aP, Worsley KJ, Poline J-P, Frith CD, Frackowiak RSJ. Statistical parametric maps in functional imaging: A general linear approach. Hum Brain Mapp. 1995; 2:189–210.

5. Fedorov A, Beichel R, Kalpathy-Cramer J, Finet J, Fillion-Robin JC, Pujol S, Bauer C, Jennings D, Fennessy F, Sonka M, Buatti J, Aylward S, Miller JV, Pieper S, Kikinis R. 3D Slicer as an image computing platform for the Quantitative Imaging Network. Magn Reson Imaging. 2012; 30:1323–1341. [PubMed: 22770690]
6. Ibanez, L., Schroeder, W., Ng, L., Cates, J. The ITK Software Guide. 2005. http://www.itk.org/ItkSoftwareGuide.pdf
7. Wolf I, Vetter M, Wegner I, Böttger T, Nolden M, Schöbinger M, Hastenteufel M, Kunert T, Meinzer HP. The medical imaging interaction toolkit. Med Image Anal. 2005; 9:594–604. [PubMed: 15896995]
8. Rex DE, Ma JQ, Toga AW. The LONI Pipeline Processing Environment. Neuroimage. 2003; 19:1033–1048. [PubMed: 12880830]
9. Lucas BC, Bogovic JA, Carass A, Bazin PL, Prince JL, Pham DL, Landman BA. The Java Image Science Toolkit (JIST) for rapid prototyping and publishing of neuroimaging software. Neuroinformatics. 2010; 8:5–17. [PubMed: 20077162]
10. Zwart NR, Pipe JG. Graphical programming interface: A development environment for MRI methods. Magn Reson Med. 2014; 74:1449–1460. [PubMed: 25385670]
11. GraphMIC. http://www.re-mic.de/index.php/graphmic
12. Bitter I, Van Uitert R, Wolf I, Ibáñez L, Kuhnigk JM. Comparison of four freely available frameworks for image processing and visualization that use ITK. IEEE Trans Vis Comput Graph. 2007:483–493. [PubMed: 17356215]
13. Gabr RE, Sun X, Pednekar A, Narayana PA. Automated patient-specific optimization of three-dimensional double inversion recovery magnetic resonance imaging. Magn Reson Med. 2016; 75:585–593. [PubMed: 25761973]
14. Bray T, Paoli J, Sperberg-McQueen CM, Maler E, Yergeau F. Extensible Markup Language (XML) 1.0 (Fifth Edition). W3C Recomm. 2008; 2008:1–38.
15. Smith SM. Fast robust automated brain extraction. Hum Brain Mapp. 2002; 17:143–155. [PubMed: 12391568]
16. Tustison NJ, Avants BB, Cook PA, Zheng Y, Egan A, Yushkevich PA, Gee JC. N4ITK: improved N3 bias correction. IEEE Trans Med Imaging. 2010; 29:1310–1320. [PubMed: 20378467]
17. Collignon A, Maes F, Delaere D, Vandermeulen D, Suetens P, Marchal G. Automated multi-modality image registration based on information theory. Inf Process Med imaging. 1995; 3:263–274.
18. Avants BB, Tustison NJ, Wu J, Cook PA, Gee JC. An open source multivariate framework for n-tissue segmentation with evaluation on public data. Neuroinformatics. 2011; 9:381–400. [PubMed: 21373993]
19. Marcus DS, Olsen TR, Ramaratnam M, Buckner RL. The Extensible Neuroimaging Archive Toolkit: an informatics platform for managing, exploring, and sharing neuroimaging data. Neuroinformatics. 2007; 5:11–34. [PubMed: 17426351]
20. Milo R, Kahana E. Multiple sclerosis: geoepidemiology, genetics and the environment. Autoimmun Rev. 2010; 9:A387–A394. [PubMed: 19932200]
21. Sahraian MA, Eshaghi A. Role of MRI in diagnosis and treatment of multiple sclerosis. Clin Neurol Neurosurg. 2010; 112:609–615. [PubMed: 20417027]
22. Sweeney EM, Shinohara RT, Shiee N, Mateen FJ, Chudgar Aa, Cuzzocreo JL, Calabresi Pa, Pham DL, Reich DS, Crainiceanu CM. OASIS is Automated Statistical Inference for Segmentation, with applications to multiple sclerosis lesion segmentation in MRI. NeuroImage Clin. 2013; 2:402–13. [PubMed: 24179794]
23. Karimaghaloo Z, Shah M, Francis SJ, Arnold DL, Collins DL, Arbel T. Automatic Detection of Gadolinium-Enhancing Multiple Sclerosis Lesions in Brain MRI Using Conditional Random Fields. IEEE Trans Med Imaging. 2012; 31:1181–1194. [PubMed: 22318484]
24. Datta S, Narayana Pa. A comprehensive approach to the segmentation of multichannel three-dimensional MR brain images in multiple sclerosis. NeuroImage Clin. 2013; 2:184–96. [PubMed: 24179773]

25. Sajja BR, Datta S, He R, Mehta M, Gupta RK, Wolinsky JS, Narayana PA. Unified approach for multiple sclerosis lesion segmentation on brain MRI. Ann Biomed Eng. 2006; 34:142–151. [PubMed: 16525763]

26. Gabr, RE., Pednekar, AS., Sun, X., Narayana, PA. IEEE-EMBS Int Conf Biomed Heal Informatics. Las Vegas, NV, USA: 2016. A Framework for Precision Magnetic Resonance Imaging: Initial Results; p. 276-279.

27. Wiggermann V, Hernandez-Torres E, Traboulsee A, Li DKB, Rauscher A. FLAIR2: A Combination of FLAIR and T2 for Improved MS Lesion Detection. Am J Neuroradiol. 2015:1–7. [PubMed: 24924551]

28. Gabr RE, Hasan KM, Haque ME, Nelson FM, Wolinsky JS, Narayana PA. Optimal combination of FLAIR and T2-weighted MRI for improved lesion contrast in multiple sclerosis. J Magn Reson Imaging [epub ahead of print]. 2016; doi: 10.1002/jmri.25281

29. Lu H, Nagae-Poetscher LM, Golay X, Lin D, Pomper M, van Zijl PCM. Routine clinical brain MRI sequences for use at 3.0 Tesla. J Magn Reson Imaging. 2005; 22:13–22. [PubMed: 15971174]

30. Polak P, Magnano C, Zivadinov R, Poloni G. 3D FLAIRED: 3D fluid attenuated inversion recovery for enhanced detection of lesions in multiple sclerosis. Magn Reson Med. 2012; 68:874–81. [PubMed: 22139997]

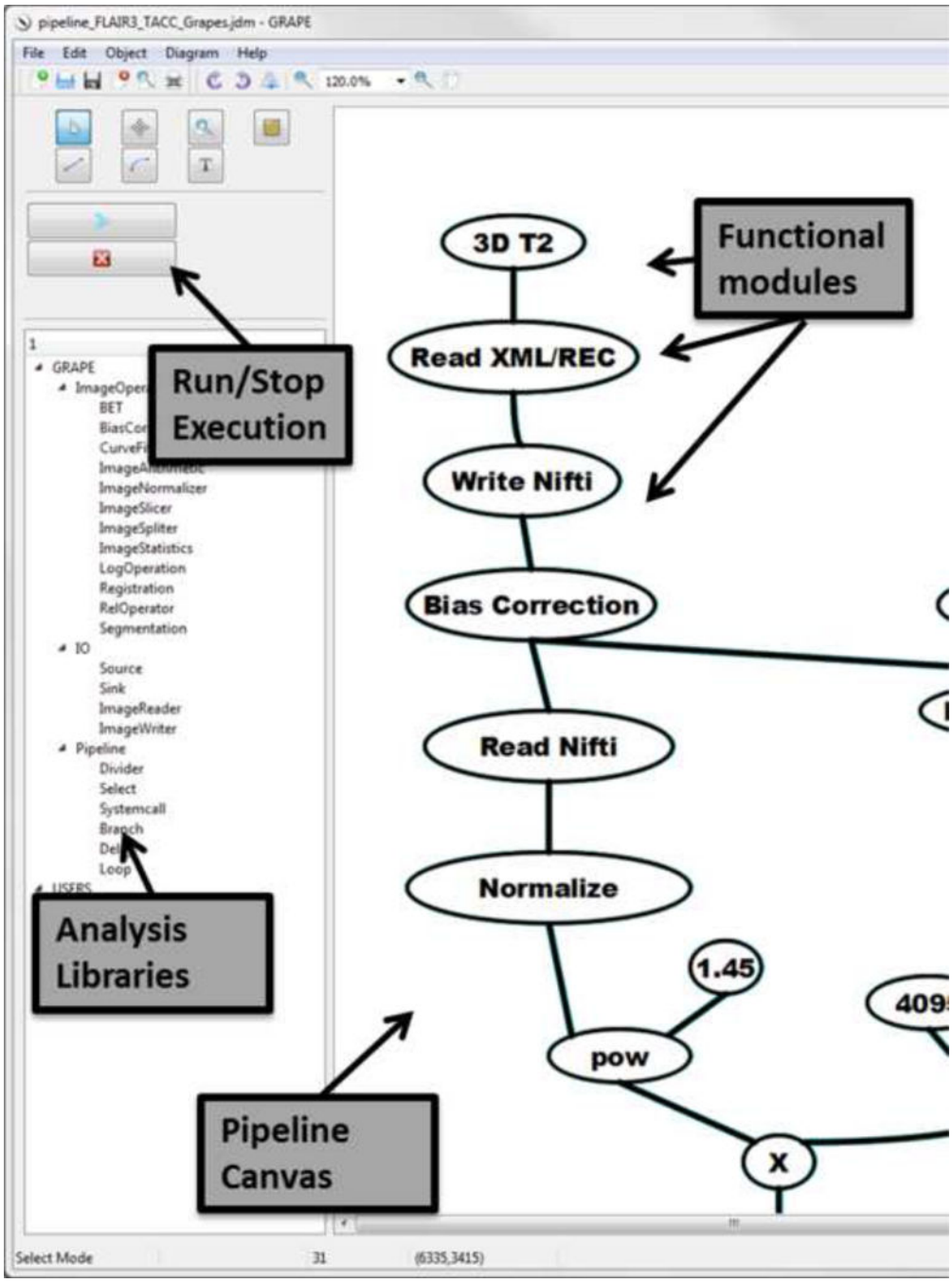

Figure 1.

Elements of the graphical user interface in GRAPE. The pipeline development toolbar (left) lists available modules and allows users to start and stop analysis. The pipeline layout panel (right) is a canvas in which users can build or view pipelines.

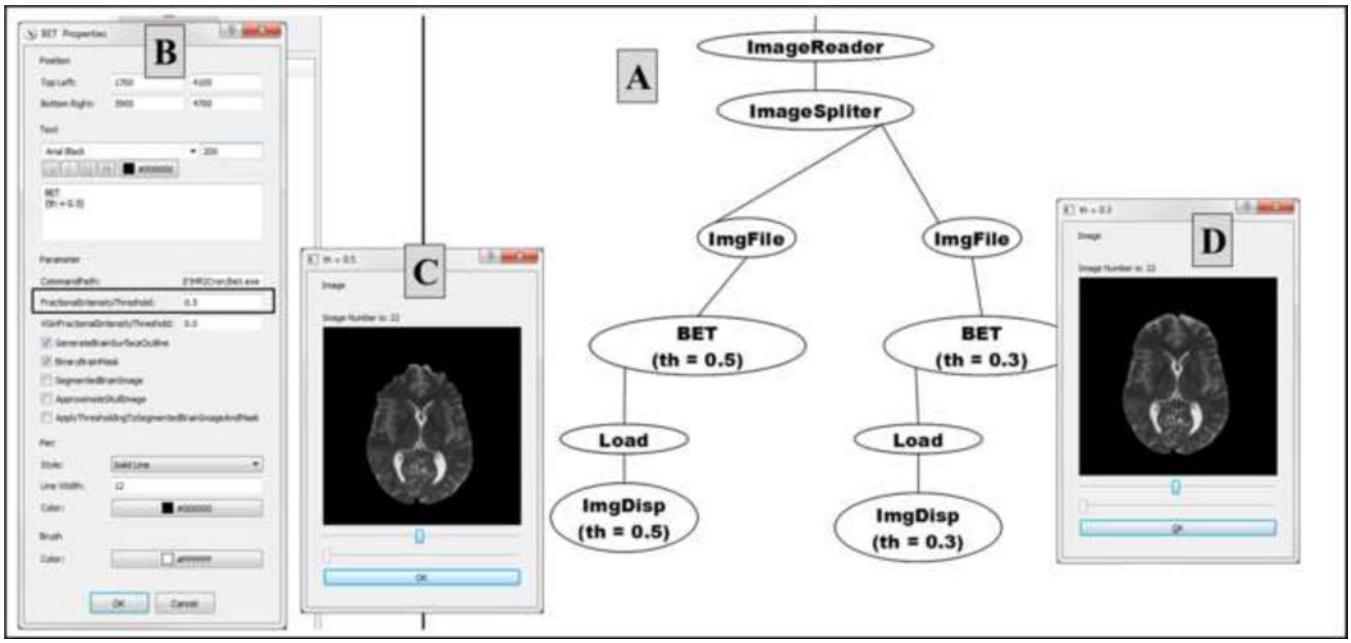

Figure 2.

This pipeline (A) compares the performance of BET with different values of the intensity threshold (th) when applied to fat-saturated T2-weighted image (acquired with a dual-echo turbo spin echo protocol). The intensity threshold was adjusted on the module property dialog (B), and the resulting images were displayed using the image display function in the Sink node, showing errors in the brain extraction done with the default threshold of 0.5 (C), compared to a better selection of the threshold at 0.3 (D).

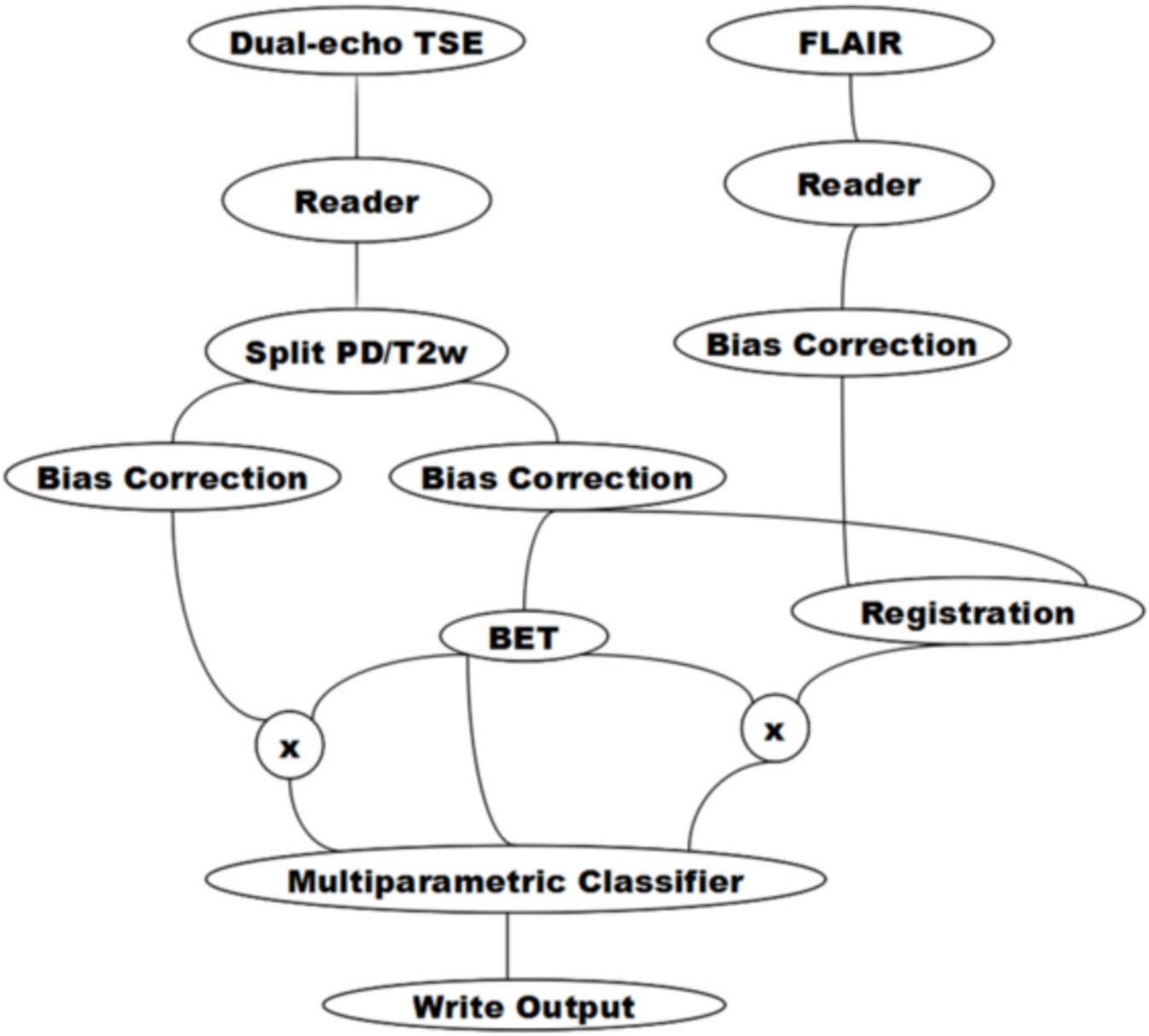

Figure 3.
Analysis pipeline for automated brain tissue classification in multiple sclerosis.

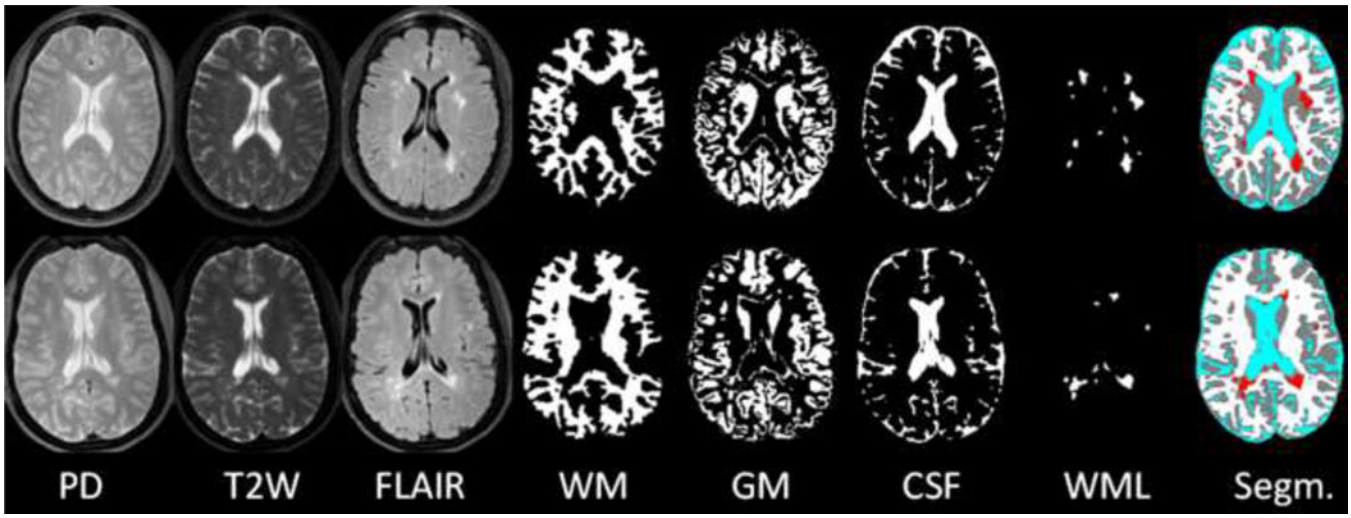

Figure 4.

Multi-parametric tissue classification obtained in 2 MS patients with an integrated analysis pipeline. In this case, GRAPE reduced data processing time by 50%.

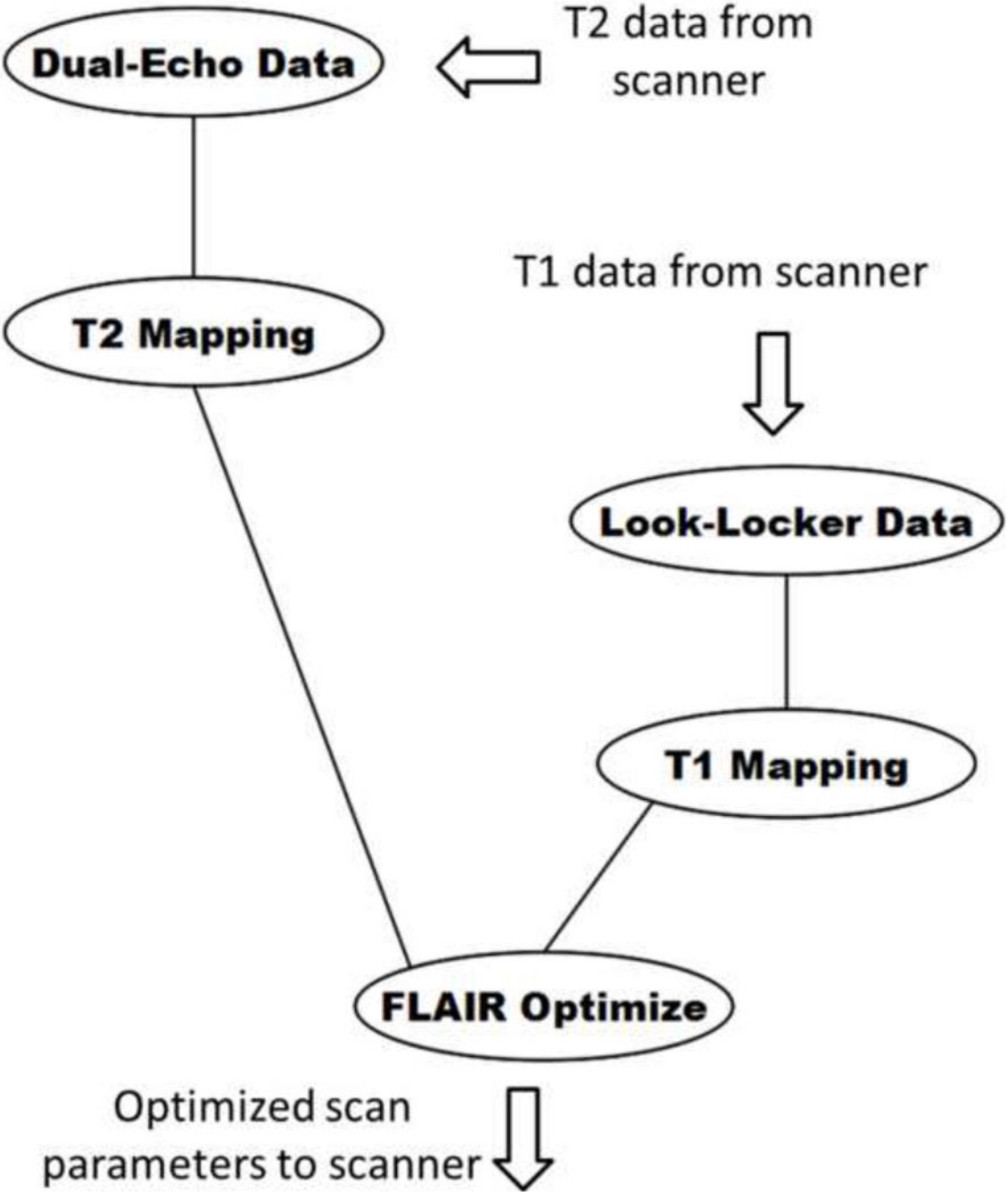

Figure 5.
The graphical pipeline for patient-specific optimization of FLAIR.

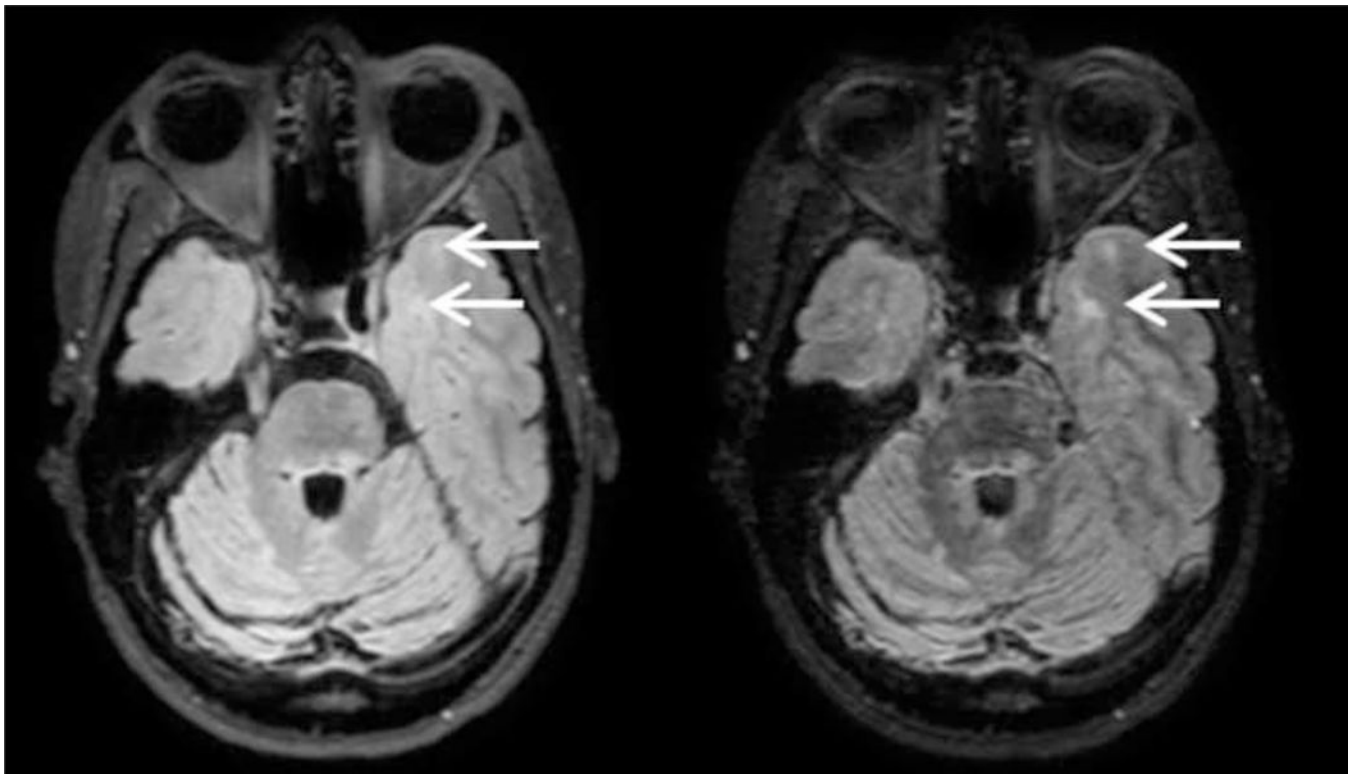

Figure 6.
The patient-specific FLAIR image (right) shows improved lesion conspicuity compared to the image acquired with the fixed FLAIR protocol (left).

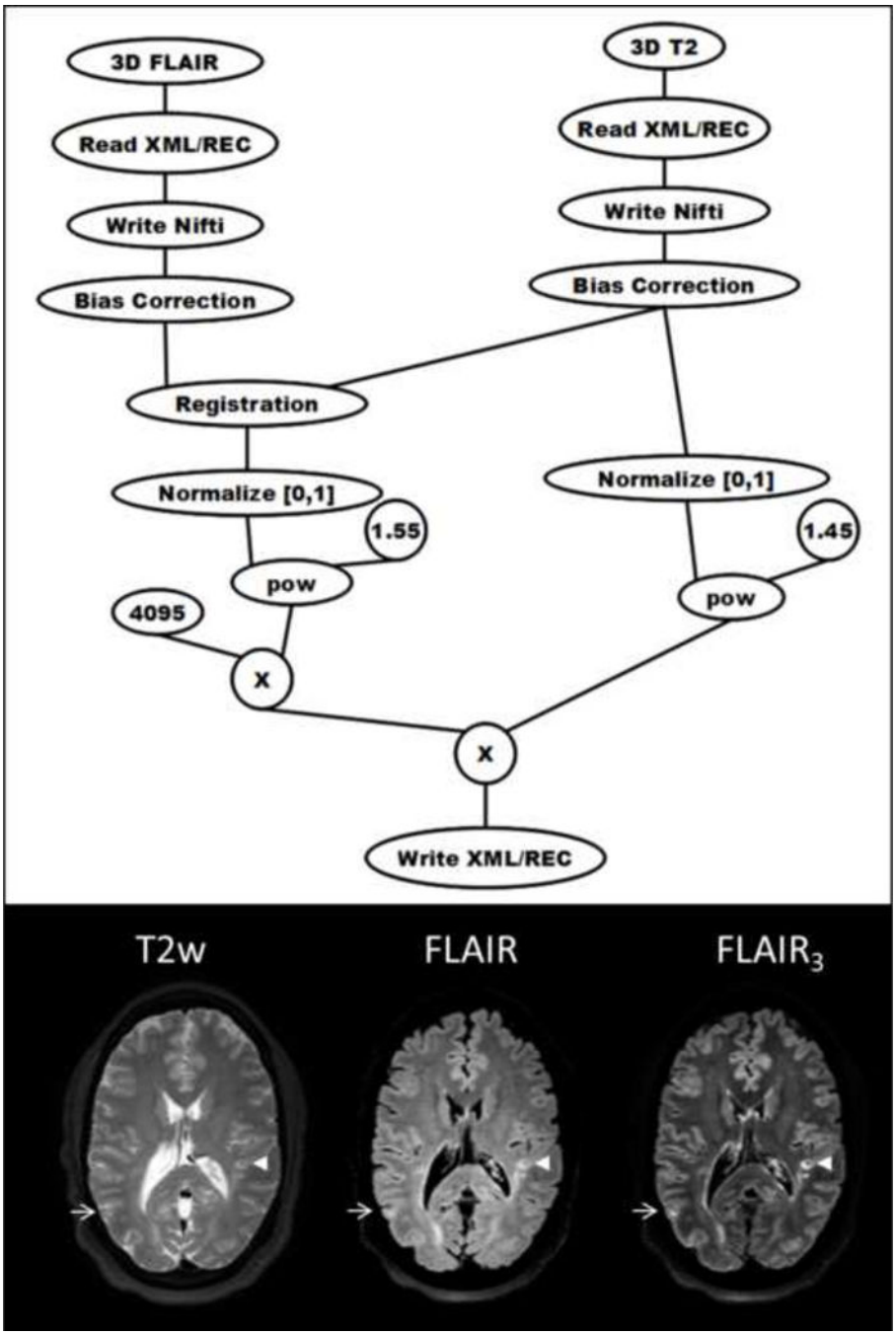

Figure 7.

Analysis pipeline of the $FLAIR_3$ method for enhancing the visualization of MS lesions (top) and representative images showing the improved lesion contrast (bottom).