Journal of Digital Imaging

Creation of DICOM—Aware Applications Using ImageJ

Daniel P. Barboriak, M.D., Anthony O. Padua, B.S., Gerald E. York, M.D., and James R. MacFall, Ph.D.

The demand for image-processing software for radiology applications has been increasing, fueled by advancements in both image-acquisition and image-analysis techniques. The utility of existing image-processing software is often limited by cost, lack of flexibility, and/or specific hardware requirements. In particular, many existing packages cannot directly utilize images formatted using the specifications in part 10 of the DICOM standard ("DICOM images"). We show how image analyses can be performed directly on DICOM images by using ImageJ, a free, Java-based image-processing package (http://rsb.info.nih.gov/ij/). We demonstrate how plug-ins written in our laboratory can be used along with the ImageJ macro script language to create flexible, low-cost, multiplatform image-processing applications that can be directed by information contained in the DICOM image header.

KEY WORDS: Image processing, image analysis, DICOM, Java

BACKGROUND

The advent of new radiological applications and the rapidity of advances in image-processing techniques have led to increasing demand for customized, flexible image-analysis software. For most research applications, two approaches are used to obtain software for image analysis: the software is either purchased from a commercial vendor or developed and produced in-house by individuals with computer programming experience. If a purchased software package can be programmed or modified, a combination of these two approaches may be used.

Packages available to perform image processing vary widely in terms of flexibility and cost. They may be designed to run on a single operating system or on multiple systems. Some packages can be implemented only on proprietary workstations or may require that images are available in particular formats.

The use of images formatted using the specifications in part 10 of the DICOM standard ("DICOM images") as a starting point for image analysis offers several advantages. First, it is often possible to obtain these images directly from imaging equipment and/or PACS storage without the need for intermediate conversion steps. Second, the DICOM headers included with these images contain valuable information, such as scan parameters in MR and CT imaging, that may be lost if images are converted to another format. The information on the header can be useful to automate analyses. For example, information on the field of view could be used to modify the size of images in the displayed output of the analysis. Assuming that the information on the DICOM header is correct, the results of the image analysis may be more reliable using this approach rather than an approach requiring users to input this information. The information on the DICOM header can also be useful for error checking, allowing errors in the scan parameters used to obtain source images to be more easily detected.

ImageJ (http://rsb.info.nih.gov/ij/) is a free, open-source image-processing platform written in Java and supported by the National Institute of Health (Wayne Rasband, National Institute of Mental Health). ImageJ runs on Windows,

From the Department of Radiology, Duke University Medical Center, PO Box 3808, Durham, NC 27710, USA.

Correspondence to: Daniel P. Barboriak, M.D., Department of Radiology, Duke University Medical Center, PO Box 3808, Durham, NC 27710, USA; tel: 919-684-7407; fax: 919-684-7157; e-mail: barbo013@mc.duke.edu

Online publication 19 April 2005
doi: 10.1007/s10278-004-1879-4

Macintosh, and UNIX platforms, and can read several image formats including RAW and DICOM part 10 format images (''DICOM images''). The ability to read DICOM images as well as the recent addition of a macro language for ImageJ has made it possible to develop low-cost image-processing applications on this platform. We describe two open source plug-ins to the ImageJ software package, Import Dicom Sequence (available at http://rsb.info.nih.gov/ij/plugins/import-dicom.html) and Query Dicom Header (available at http://rsb.info.nih.gov/ij/plugins/query-header.html) developed in our laboratory to simplify the utilization of DICOM images in ImageJ. We then demonstrate the utility of this approach by describing three applications that were implemented as ImageJ macro scripts.

METHODS

Plug-in Development

Import DICOM Sequence

ImageJ shares with several other image-analysis software programs the capability to open and display DICOM images. Image analysis using single images can be easily performed using these native capabilities. ImageJ can also perform analyses on multiple images within a DICOM series: as currently configured, all the DICOM images in a directory can be imported into a "stack" of images.

The order in which DICOM image files are imported is crucial for efficient data processing. It is important to note that there is no "correct" order for sorting: the most appropriate order for sorting depends upon the application for which the images are to be used. For many applications, the most appropriate order is an anatomic one (inferior to superior slice position, for example). On the other hand, a dynamic contrast-enhanced sequence may be most logically sorted primarily by image acquisition time and secondarily by image location.

One limitation of ImageJ is that DICOM images are imported in order of the title of the DICOM image file. This feature, common to many image-analysis packages, is problematic because the titles of DICOM image files are not specified as part of the DICOM standard. Instead, each PACS and imaging equipment vendor is free to adopt its own convention in assigning filenames to DICOM image file. As a result, schemes used to assign titles to DICOM image files vary greatly between and even within PACS and imaging equipment vendors. Importantly, there is no guarantee that importing images in the order of DICOM image file title will result in stack of images ordered in a clinically useful way.

A common reason for poorly ordered series is that number suffixes on image titles are frequently not padded with zeros. For example, ImageJ would import a file titled "image10" after "image1" but before "image2" (Fig 1). This problem can be solved by padding the numeric component of image titles with leading zeros to make titles of equal length.

Other problems are less easily addressed. For example, for some vendors, image titles may be assigned in the order in

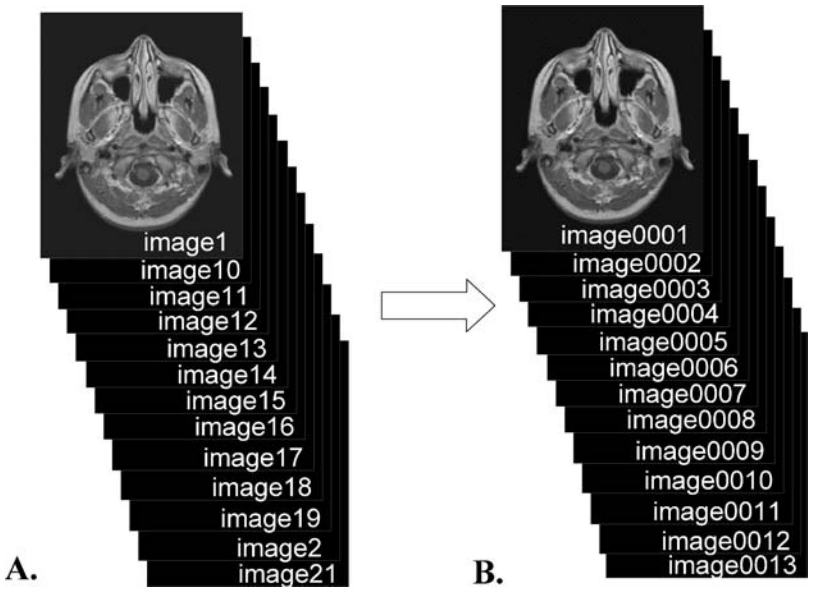


Fig 1. Importing a series of images in order of image title may lead to poorly ordered image stacks. A. When the character-equivalent of the image title is used, the title "image10" sorts between "image1" and "image2." B. This problem is frequently solved by appending leading zeros to the numeric portion of the image title. The Import Dicom Sequence plug-in uses this solution when image number information (group:element 0020:0013) is not available in the DICOM header.

which the image was acquired. Although this order may often be useful, it is generally not ideal when images are not obtained in anatomic order (as would be the case, for example, when an interleaved MR imaging sequence is performed).

We created the Import Dicom Sequence plug-in to help address these problems. The plug-in modifies ImageJ's image sequence importing capability by sorting the images to import them in order of image number in the DICOM header (group:element 0020:0013). If no image number is available in the header, the numeric component of the image title is padded with leading zeros as described above and the images are imported in order of the modified title.

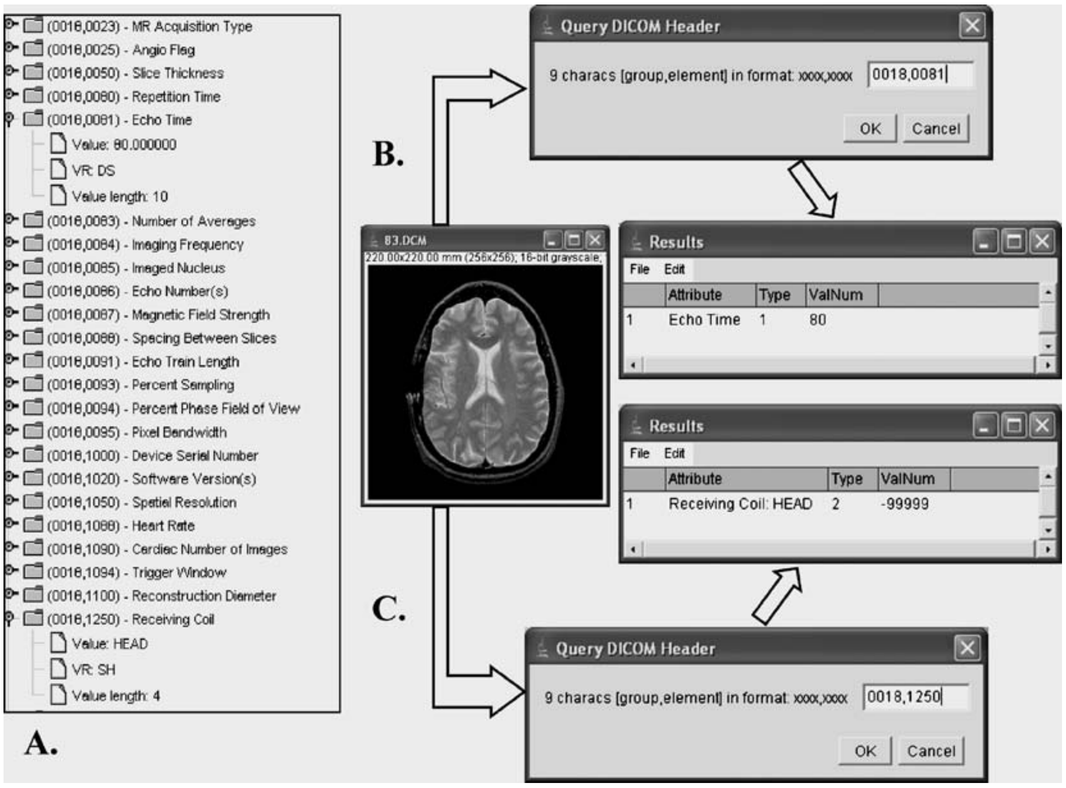
One limitation of this approach is that only sorting by image number is supported. This open-source plug-in can be modified, however, to sort by other fields in the DICOM header.

Query DICOM Header

The development of a macro scripting language for ImageJ allows the creation of flexible, sophisticated sequences of image-processing operations that can be saved and repeated. Macro scripts can be created by recording operations from the ImageJ toolbar, by writing scripts using ImageJ's reasonably simple macro language, or both. Macro scripting in ImageJ is a powerful way to create applications using a variety of image-processing functions already prebuilt into ImageJ as well as user-contributed plug-ins, many of which can be used within macro scripts.

Although ImageJ allows users to view DICOM header information within a text window using the "Show Info" command, the information in this form cannot be easily used to direct the actions of a macro script. In particular, extracting the value of a particular DICOM tag is unwieldy using this approach.

The Query Dicom Header plug-in is designed to retrieve data from the DICOM header in a form convenient for use in macro scripts. The user or script provides the group and element code of the DICOM tag, and the value for that tag is returned in ImageJ's results table (Fig 2). The plug-in also returns a value called "Type" which specifies whether the value for the field in the DICOM header is numeric (Type equals 1), nonnumeric (Type equals 2), or missing or invalid (Type equals 9). The value for the field in the DICOM header can be extracted within a macro script using the *getResult* function if the result is numeric or the *getInfo* function if the result is nonnumeric, and these data can be used to direct further image processing.



```
(0018,0023) - MR Acquisition Type
(0018,0025) - Angio Flag
(0018,0050) - Slice Thickness
(0018,0080) - Repetition Time
(0018,0081) - Echo Time
    Value: 80.000000
    VR: DS
    Value length: 10
(0018,0083) - Number of Averages
(0018,0084) - Imaging Frequency
(0018,0085) - Imaged Nucleus
(0018,0086) - Echo Number(s)
(0018,0087) - Magnetic Field Strength
(0018,0088) - Spacing Between Slices
(0018,0091) - Echo Train Length
(0018,0093) - Percent Sampling
(0018,0094) - Percent Phase Field of View
(0018,0095) - Pixel Bandwidth
(0018,1000) - Device Serial Number
(0018,1020) - Software Version(s)
(0018,1050) - Spatial Resolution
(0018,1088) - Heart Rate
(0018,1090) - Cardiac Number of Images
(0018,1094) - Trigger Window
(0018,1100) - Reconstruction Diameter
(0018,1250) - Receiving Coil
    Value: HEAD
    VR: SH
    Value length: 4
```

A.

Query DICOM Header

9 characs [group,element] in format: xxxx,xxxx 0018,0081

OK Cancel

B.

83.DCM

220.00x220.00 mm (256x256); 16-bit grayscale;

Results

File Edit

	Attribute	Type	ValNum
1	Echo Time	1	80

Results

File Edit

	Attribute	Type	ValNum
1	Receiving Coil: HEAD	2	-99999

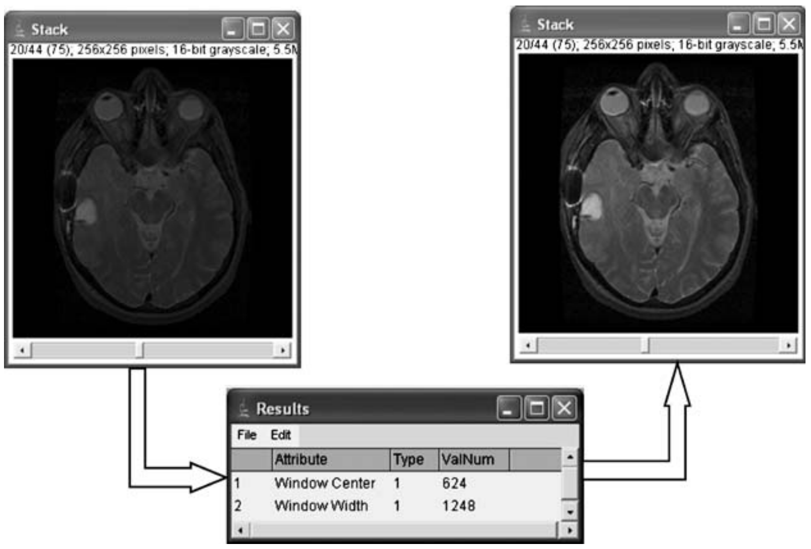
C.

Query DICOM Header

9 characs [group,element] in format: xxxx,xxxx 0018,1250

OK Cancel

Fig 2. Use of Query Dicom Header to extract data from the DICOM header. A. A portion of the DICOM header from an image is displayed. Two entries, Echo Time and Receiving Coil are expanded for display. (Display from NeoLogica Dicom Dumper, v.1.1, www.neologica.it). The corresponding image is opened in ImageJ, and Query Dicom Header is used to query the Echo Time (B) and Receiving Coil (C). The result of the query is placed in ImageJ's Results table. Type is coded 1 for numeric data, 2 for text data, and 9 for missing or invalid data. Query Dicom Header can also be triggered within a macro, and the result of the query retrieved within the macro from the results table.



	Attribute	Type	ValNum
1	Window Center	1	624
2	Window Width	1	1248

Fig 3. First application: windowing and leveling based on DICOM header fields. Dual echo spin-echo T2-weighted images are imported into a stack using the Import Dicom Sequence plug-in. The macro script used the Query Dicom Header plug-in to query the Window Center and Window Width fields in the DICOM header, and automatically sets the minimum and maximum signal intensities displayed in ImageJ based on this information.

One potential limitation* of this technique is that each DICOM image file is assumed to be composed of a single image and associated header data. Although current implementations of the DICOM image format generally meet this assumption, support will be needed in the future for the recently announced second-generation "DICOM Image Object Definitions for MR images,"[1] which will use a multiframe image format.

RESULTS

We used the Import Dicom Sequence and Query Dicom Header plug-ins to produce three example applications using ImageJ's macro language features. All three macro scripts are available at http://www.radweb.mc.duke.edu/dbplab/SCAR2004.html.

Auto Window/level

The first application uses the Query Dicom Header plug-in within the macro script to query and extract the values from the Window Center (group:element 0028:1050) and Window Width (group:element 0028:1051) fields in the header of the active image window in ImageJ, then adjusts the display of the image accordingly (Fig 3). This simple script is shown in its entirety (Fig 4).

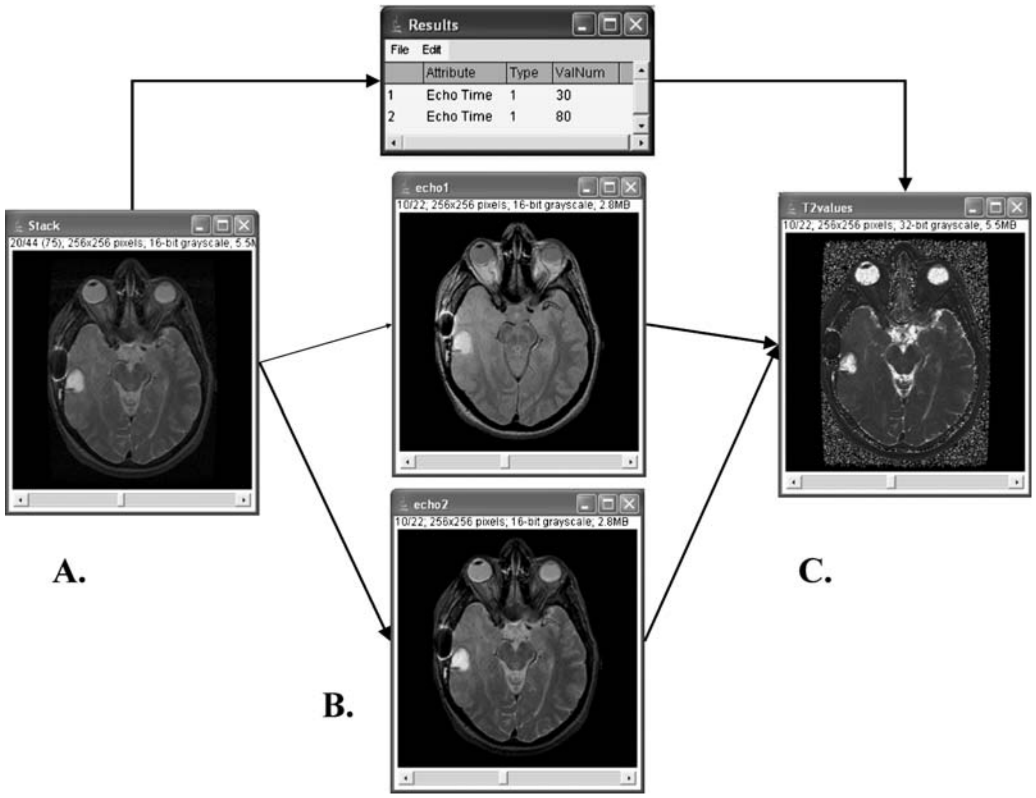
* The tag-related limitation mentioned at the beginning of this paragraph was fixed on December 4, 2004 and is no longer operative.

T2 Parameter Maps

The second application is designed to generate maps of estimated T2 from dual-echo spin-echo T2-weighted MR images (Fig 5). This approach, which assumes mono-exponential T2 decay in the underlying structures, has been used to character-

```
run("Clear Results");
run("Set Measurements...", "decimal=3");
run("Query Dicom Header", "9=0028,1050");
run("Query Dicom Header", "9=0028,1051");
typechk = getResult("Type",0);
if (typechk == 9) exit;
typechk = getResult("Type",1);
if (typechk == 9) exit;
center = getResult("ValNum",0);
width = getResult("ValNum",1);
setMinAndMax(center-(width/2), center+(width/2));
```

Fig 4. Macro script used to perform first application. The script clears the results table, runs the Query Dicom Header plug-in from within the macro script to extract the Window Center (group:element 0028:1050) and Window Width (group: element 0028:1051) fields in the DICOM header, performs error checking to ensure the values for these fields are present and valid, extracts the values from the results table, and uses these values to set the minimum and maximum pixel intensities displayed.


Results
File Edit
Stack
20/44 (75); 256x256 pixels; 16-bit grayscale; 5.5M
echo1
10/22; 256x256 pixels; 16-bit grayscale; 2.8MB
echo2
10/22; 256x256 pixels; 16-bit grayscale; 2.8MB
T2values
10/22; 256x256 pixels; 32-bit grayscale; 5.5MB
A.
B.
C.

	Attribute	Type	ValNum
1	Echo Time	1	30
2	Echo Time	1	80

Fig 5. Second application: T2 parameter maps generated from dual echo spin-echo T2 weighted images. A. Dual echo spin-echo T2-weighted images are imported into a stack using the Import Dicom Sequence plug-in. B. The macro script uses the Query Dicom Header plug-in to extract Echo Time fields (group:element 0018:0081) from the DICOM header, and divides the stack into two substacks representing the long TR long TE and long TR short TE images respectively. C. The T2 parametric map is calculated using a standard formula by performing image mathematics on the two substacks and multiplying by the difference in echo times.

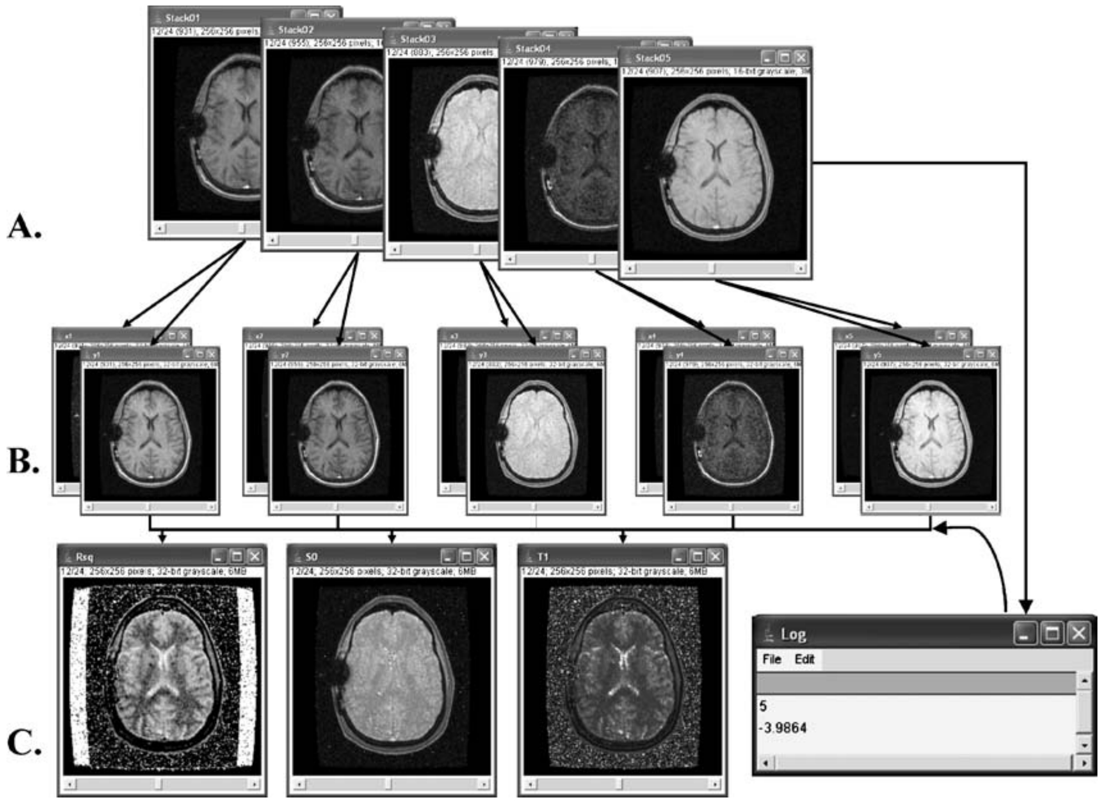
ize hippocampal abnormalities in patients with mesial temporal sclerosis.[2,3]

Four steps are used to generate the T2 maps:

1. The Import Dicom Sequence plug-in is used to import DICOM images from a dual-echo spin-echo T2-weighted sequence into a stack. At our institution, these images opened in image number order will create a stack of alternating long TR short TE and long TR long TE images from the same location.
2. The Query Dicom Header is used to extract the echo times (group:element 0018:0081) from the first two images, assuming that the echo times are identical for every image pair in the series.
3. A plug-in called Substack Maker is used to separate the stack into two substacks, the first containing long TR short TE images (odd numbered images), and the second containing long TR long TE images (even numbered images).
4. The T2 maps are calculated using the standard formula $T2 = (TE_2 - TE_1)/\ln(SI_1/SI_2)$, where TE_i is the echo time from the ith echo, and SI_i is the signal intensity from the ith echo. This step illustrates how ImageJ's mathematics capabilities can be used to add, subtract, multiply, or divide images by constants or variables within a macro script. In addition, images or image stacks can be added, subtracted, multiplied, or divided by each other. Logarithms and reciprocals of images and image stacks can also be obtained.

T1 Parameter Maps

The third application is designed to generate maps of T1 and S_0 (equilibrium magnetization) from T1-weighted spoiled gradient echo (SPGR) images acquired using multiple flip angles using a linear fit solution (Fig 6). Mapping of tissue T1 is



```
5
-3.9864
```

Fig 6. Third application: T1 parameter maps generated from multiple 3D spoiled gradient echo in steady state (SPGR) images using variable flip angles. A. In this example, five series of 3D SPGR images are imported into separate stacks using the Import Dicom Sequence plug-in. B. For the purposes of performing a linear correlation between images, each of the five stacks is converted to a pair of ordinate and coordinate stacks using Flip Angle information obtained using Query Dicom Header. C. The slope and intercept stacks obtained from the linear correlation of ordinate and coordinate stacks (not shown), along with information derived from the DICOM header placed in the log table (the number of pairs and the repetition time obtained using Query Dicom Header are listed), are used to calculate stacks of 32-bit quantitative maps of linear correlations (Rsq), equilibrium magnetization (S0) and T1.

used as a preparatory step for measuring contrast permeability using dynamic contrast-enhanced T1-weighted imaging.[4,5] For SPGR images, the relationship between the signal intensity obtained at a given flip angle $S(\alpha)$ and the S_0 and T1 is expressed by the equation $S(\alpha) = S_0\,(1 - e^{-TR/T1})\sin(\alpha)/(1 - \cos(\alpha)e^{-TR/T1})$, where TR is the repetition time. Given the signal intensity data at a number of known flip angles, this equation can be solved for T1 and S_0 using nonlinear fitting. An alternative approach is to linearize the equation by using trigonometry and rearranging terms, and to solve for T1 and S_0 from the slope and intercept. The equation can be rewritten as $S(\alpha)/\sin(\alpha) = mS(\alpha)/\tan(\alpha) + S_0\,(1 - m)$, where $m = e^{-TR/T1}$. This equation is now in a linear form such that a plot of $S(\alpha)/\sin(\alpha)$ *versus* $S(\alpha)/\tan(\alpha)$ has a slope of m and an intercept of $S_0(1 - m)$.[6] T1 and S_0 can be calculated directly from the slope and intercept.

The macro script application calculates T1 maps in three steps:

1. For this application, we assumed that the SPGR images obtained at each flip angle are placed in separate directories. The user inputs the number of flip angles to be used in the mapping, and the macro script then uses the *getDirectory* macro function to allow the user to specify the location of each directory through a graphical user interface. Each SPGR sequence is opened as a separate stack using the Import Dicom Sequence plug-in.
2. The Query Dicom Header plug-in is used within the macro script to extract each flip angle (group:element 0018:1314) and the rep-

etition time (group:element 0018:0080) for the stacks. Ordinate and coordinate stacks are produced for linear fitting by dividing each flip angle stack by the sin(α) and tan(α), respectively.

3. The formula for linear regression is applied to the ordinate and coordinate stacks to perform a least squares linear fit and solve for the slope, intercept, and correlation. The stacks corresponding to the slope and intercept are used to produce pixel-by-pixel maps of T1 and S_0.

DISCUSSION

Low-cost Platform for Development of Image-processing Applications

When a research laboratory decides to use image-analysis software for research purposes, several fundamental decisions must be made. First, would development of software be feasible as an in-house project, or is there off-the-shelf technology available to do the task? How much of the solution can be purchased, and how much should be produced? The answer will depend to a large extent on local factors: familiarity with image-analysis platforms, flexibility of the platform, local support available, local programming resources available, format of images, and hardware resources (such as workstations) available.

There are many image-processing software packages available, each with different features. Although it is unrealistic to expect a single software package to provide an ideal solution to all image-analysis problems, the profusion of software has some undesirable consequences. Although several software packages may perform the same type of image processing, differences in the approach and in the coding of the software in each package may affect the results of the analysis. As a result, differences in image-analysis software may make it more difficult to compare the results obtained in one lab with those obtained in another. This is particularly of concern because the extent to which software packages are evaluated for reproducibility and bias using test data is quite variable. Even when testing uncovers programming errors, software packages, especially proprietary products, may be difficult to quickly repair.

The ideal software solution would be inexpensive, flexible, run on multiple operating systems, and support automated analyses. In addition, the ideal software should also be able to use DICOM images as these are often readily available and contain valuable information in the DICOM header.

In this article, we suggest that a programmable, open-source, Java-based software solution offers many advantages and illustrate this approach by developing sample applications of varying complexity using ImageJ's macro language. This macro language works synergistically with the Import Dicom Sequence and Query Dicom Header plug-ins to allow the development of flexible DICOM-aware image-analysis applications.

Advantages

Assuming that the information in the DICOM header is accurate, the adoption of a DICOM-aware approach to image analysis offers several advantages. The most important of these is that the information in the DICOM header can be used to customize analyses. In this context, it is important to keep in mind that conversion of imaging data to other image formats as is required by some image-analysis software generally leads to loss of much of the information in the DICOM header. Another advantage is that error checking for complicated imaging protocols becomes reasonably straightforward: the image headers can be queried to ensure that image data were obtained using the proper scanning parameters.

The use of ImageJ's macro language to create DICOM-aware applications facilitates the development of automated image analyses, an approach that saves time when repeated analyses are performed. By reducing errors associated with manual input of scan parameters, this approach would additionally be expected to improve reproducibility.

This approach will only be widely applicable if the image-processing functions provided by the software are reasonably broad and robust. In ImageJ, there is native support for many image-processing functions beyond simple image arithmetic, including smoothing, sharpening, edge detection, masking, binary operations, Fourier transformations, and watershed transformations. In addition, there are over 200 user-supplied plug-ins listed on the ImageJ site, many of which can be called from within a macro script. These plug-ins support a wide variety of image-processing functions; for example, addition of image noise, spatial transformation of image stacks, segmentation by *k*-means clustering, and specialized edge detection. Additionally, Java-based libraries can be incorporated into ImageJ to provide expanded functionality. For example, matrix algebra packages such as Jama (http://math.nist.gov/javanumerics/jama/, National Institute of Standards and Technology) facilitate the use of elementary matrix operations within plug-ins. The power of macro applications can also be expanded by incorporating ImageJ into image-processing pipelines such as the LONI pipeline (UCLA Laboratory of Neuroimaging, http://www.loni.ucla.edu/ICBM/ICBM_Pipeline.html),[7] which allow several software packages to interact with imaging data.

The many advantages of using ImageJ as the platform to create image-processing applications are enhanced by ImageJ's use of the Java programming language and its open source code. By using Java, ImageJ can be implemented on computers using a variety of operating systems, generating a cost savings compared with software requiring proprietary and expensive workstations. By using an open-source approach, software tools developed on ImageJ are often more "repairable" and "evolvable" than software produced with proprietary techniques. An open-source approach also allows faster implementation of new image-processing algorithms as they become available. A key component of this approach is the development of a community of interested users and developers to provide feedback and stimulate innovation. Over a period of years, ImageJ has fostered an international users group which communicates via a mailing list (http://rsb.info.nih.gov/ij/list.html).

Limitations

Although the number of image-processing functions and plug-ins available when using ImageJ's macro language allows a wide variety of image-processing problems to be addressed, there are several practical limitations that should be kept in mind when using this approach. Perhaps most importantly, pixel-by-pixel processing is performed approximately 100 times slower in a macro script than in a plug-in (http://list.nih. gov/cgi-bin/wa?A2=ind0103&L=imagej&P= R2426). As a rule, when pixel-by-pixel processing is needed, we use plug-ins written in Java rather than macro scripts to perform these operations. Many functions that create new images (e.g., image mathematics) use a Java plug-in structure that can be called from within macro scripts to handle image creation with no significant loss of speed. On the other hand, macro scripts can be useful as a preliminary step or "mock up" to test the feasibility of creating a Java plug-in. We have found little difference between the speed of a plug-in that calls a series of plug-ins and the speed of a macro script that calls the same series of plug-ins.

Although plug-ins are available that allow DICOM images to be created, the ability of ImageJ to correctly create DICOM tags with appropriate values for these images is limited. For this reason, there is no easy way to upload the quantitative images such as those created in our example second and third applications to a PACS system. There is currently no support in ImageJ for reading or creating DICOMDIR files, which would allow more convenient access to a series of images.

This approach is also limited in several respects by limitations in the macro language itself. First, error handling is much more primitive in a macro script than within a Java plug-in. Second, the graphical user interface for either entering data or receiving text output from a macro script is relatively primitive. Finally, passage of data between plug-ins and macro scripts can be awkward. The macro language has been increasing in sophistication since it has been introduced, and further improvements in its functionality are expected.

CONCLUSION

We have demonstrated the utility of using ImageJ's macro language along with plug-ins produced in our laboratory to produce low-cost, automatable, DICOM-aware image-processing applications. It should be noted in closing that we would anticipate that these applications are most appropriate for use in a research setting. Software applications used for diagnostic purposes in a clinical setting are considered medical devices and are generally subject to regulatory approval from authorities such as the Food and Drug Administration (in the United States), requiring rigorous verification and validation procedures for this purpose.

ACKNOWLEDGMENT

The authors gratefully acknowledge the support of a Seed Grant from the Radiological of North America. The support of ImageJ software by Wayne Rasband and the ImageJ users and developers group is also acknowledged with gratitude.

REFERENCES

1. Clunie D, Parisot C: New enhanced multiframe DICOM CT and MR objects to enhance performance and image processing on PACS and workstations. The 21st Meeting of the Society for Computer Applications in Radiology, Vancouver, British Columbia, Canada, p 88, 2004

2. Duncan JS, Bartlett P, Barker GJ: Technique for measuring hippocampal T2 relaxation time. Am J Neuroradiol 17:1805–1810, 1996

3. Duncan JS, Bartlett P, Barker GJ: Measurement of hippocampal T2 in epilepsy. Am J Neuroradiol 18:1791–1792, 1997

4. Evelhoch JL: Key factors in the acquisition of contrast kinetic data for oncology. J Magn Reson Imaging 10:254–259, 1999

5. Li KL, Zhu XP, Waterton J, Jackson A: Improved 3D quantitative mapping of blood volume and endothelial permeability in brain tumors. J Magn Reson Imaging 12:347–357, 2000

6. Bluml S, Schad LR, Stepanow B, Lorenz WJ: Spin-lattice relaxation time measurement by means of a TurboFLASH technique. Magn Reson Med 30:289–295, 1993

7. Rex DE, Ma JQ, Toga AW: The LONI pipeline processing environment. NeuroImage 19:1033–1048, 2003