IceLuva: a scripting engine for fast development of reconstruction algorithms

F. Santini[1], S. Patil[1,2], and K. Scheffler[1]
[1]Radiological Physics, University of Basel Hospital, Basel, Basel, Switzerland, [2]Center for Applied Medical Imaging, Siemens Corporation, Corporate Research, Baltimore, MD, United States

Introduction. Reconstruction of MRI images is a process that normally involves many more steps rather than simple two-dimensional fast Fourier transform (FFT), as predicted in theory. Manipulation of the k-space is needed in order to apply acceleration algorithms (partial Fourier acquisition, parallel imaging), image space manipulation is used to derive quantitative and qualitative parameters from set of images (T1, T2, apparent diffusion coefficient, magnetization transfer…), and filtering can be applied either in k-space or in image space. MR scanner manufacturers offer the possibility to greatly customize the reconstruction process, by tweaking the predefined reconstruction parameters, or by writing custom reconstruction software to be run on the dedicated computer bundled with the scanner. However, the programming framework varies greatly among the manufacturers, and it is often complex and tedious to write a reconstruction program that performs even simple operations, with the result that researchers resort to offline export and image manipulation on an external computer. In this work, we present a piece of software that integrates into the reconstruction framework of the scanner and allows rapid development of image postprocessing steps. This software is based on Lua [1] and other open source code released with permissive licenses (MIT/X and BSD-like), thus allowing easy porting among different platforms and the possibility of closing the source afterwards.

Materials and methods. The software was developed in C/C++ for the Siemens Image Calculation Environment (ICE), and consists of an object ("functor") connected to the end of the reconstruction pipeline that temporarily stores the images before they are sent to the scanner console. A Lua interpreter is embedded into the functor, and an external script, specified in the sequence interface, is loaded. The script interacts with the environment to retrieve image data and forward it to the scanner console and patient database. A custom extension of the Lua language provides a syntax for image manipulation similar to the one of Matlab (The Mathworks, Natick, MA), automating functions like algebraic manipulation nonlinear fitting, registration, and so forth. Scripts were developed for a number of real-world situations, ranging in complexity from simple manipulation of contrast to pixelwise non-linear fitting of T1 and T2 quantification, to calculation of phase-contrast images from large three-dimensional, high-resolution datasets. The framework was evaluated under two aspects: implementation simplicity and computational efficiency. The former parameter was assessed by calculating the number of logical lines of code (LLOC) of three example scripts (contrast inversion, T2 fitting, phase contrast), while the latter was determined by the total running time of the scripts compared to equivalent operations implemented in Matlab.

Results. The contrast inversion script consisted of 4 LLOC (fig 1), the T2 fitting script using non-linear fitting consisted of 38 LLOC, and a more complex example, for calculation of phase contrast images that can be used for through-plane, in-plane, or three-directional calculation consisted of 110 LLOC. The Lua reconstruction outperformed Matlab in all tasks, the difference being most remarkable when a large number of images were involved, because of disk operations for image retrieval in the offline case (table 1).

Discussion. The IceLuva functor allowed rapid and intuitive development of scripts that could be run directly on the reconstruction computer of the scanner, thus avoiding most the need to export data offline for postprocessing on a separate computer. The performance of the algorithms was always superior to offline processing in Matlab, but inherently less efficient than a pure native implementation, due to the overhead of the interpreter. The permissive MIT/X and BSD licenses allowed the integration of the open source Lua code with the proprietary code of the manufacturer, covered by non-disclosure agreements, and at the same time would facilitate the porting of the code to different platforms, providing an abstraction layer for the development of cross-platform reconstruction programs.

	Processing time (ms)	
	IceLuva	Offline processing
Contrast inversion	210	8491
T2 fitting	4490	14160
Phase contrast	7590	38493

Table 1: Processing times for three reconstruction algorithms on representative datasets.

```
for imageID in ice.allImages() do
        img=getImage(imageID); -- retrieve image
        storeImage(imageStamp, -- store image
                img,
                1,
                "Original image"); -- image comment
        storeImage(imageStamp,
                ice.max(img) - img, -- invert contrast
                2, -- store as different DICOM series
                "Inverted image"); -- image comment
end
```

Fig. 1: Code for the contrast inverter script

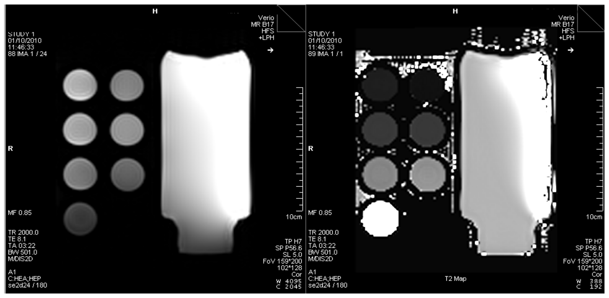

Fig 2:Original image series and T2 map loaded in the scanner console

References. [1] Ierusalimschy R, *et al.*, Lua - an extensible extension language, Software: Practice & Experience 26 #6 (1996) 635–652