Computers in Biology and Medicine 58 (2015) 56–62

Contents lists available at ScienceDirect

Computers in Biology and Medicine

journal homepage: www.elsevier.com/locate/cbm

Development and validation of an open source quantification tool for DSC-MRI studies

P.M. Gordaliza [a,b,*,1], J.M. Mateos-Pérez [a,c,1], P. Montesinos [a,b], J.A. Guzmán-de-Villoria [d], M. Desco [a,b,c], J.J. Vaquero [a,b,*]

[a] Instituto de Investigación Sanitaria Gregorio Marañón, Madrid, Spain
[b] Departamento de Bioingeniería e Ingeniería Aeroespacial, Universidad Carlos III de Madrid, Madrid, Spain
[c] Centro de Investigación Biomédica en Red de Salud Mental (CIBERSAM), Madrid, Spain
[d] Servicio de Radiodiagnóstico, Hospital General Universitario Gregorio Marañón, Madrid, Spain

ARTICLE INFO

Article history:
Received 22 September 2014
Accepted 1 January 2015

Keywords:
Magnetic resonance imaging
Perfusion
ImageJ
Open source
Software tool

ABSTRACT

Motivation: This work presents the development of an open source tool for the quantification of dynamic susceptibility-weighted contrast-enhanced (DSC) perfusion studies. The development of this tool is motivated by the lack of open source tools implemented on open platforms to allow external developers to implement their own quantification methods easily and without the need of paying for a development license.

Materials and methods: This quantification tool was developed as a plugin for the ImageJ image analysis platform using the Java programming language. A modular approach was used in the implementation of the components, in such a way that the addition of new methods can be done without breaking any of the existing functionalities. For the validation process, images from seven patients with brain tumors were acquired and quantified with the presented tool and with a widely used clinical software package. The resulting perfusion parameters were then compared.

Results: Perfusion parameters and the corresponding parametric images were obtained. When no gamma-fitting is used, an excellent agreement with the tool used as a gold-standard was obtained ($R^2 > 0.8$ and values are within 95% CI limits in Bland–Altman plots).

Conclusion: An open source tool that performs quantification of perfusion studies using magnetic resonance imaging has been developed and validated using a clinical software package. It works as an ImageJ plugin and the source code has been published with an open source license.

1. Introduction

Dynamic susceptibility-weighted contrast-enhanced (DSC) perfusion studies in magnetic resonance imaging (MRI) provide valuable data for brain function research and clinical practice. This image modality is based on the analysis of signal intensity changes in the MRI signal following the intravenous injection of a bolus of a paramagnetic contrast agent, such as Gd-DTPA [24]. When the bolus passes through the brain, the signal intensity drops on T2*-weighted images due to small variations in the local magnetic field. Modeling the time course of this tracer through the brain tissue makes it possible to obtain functional information regarding perfusion-related parameters such as cerebral blood flow (CBF), mean transit time (MTT) and cerebral blood volume (CBV).

The correct quantification of these parameters has several clinical applications, such as detection and assessment of ischemic stroke prior to treatment [3], characterization of multiple sclerosis lesions [7], tumor diagnosis [1,13,4,12,9,6] or as indicators on the progress of Alzheimer's disease [8]. As this technique is also widely used in preclinical studies [20,25,26,15], it is therefore interesting to have a tool that performs the quantification process in a fast and reliable way for research purposes. To the extent of our knowledge, the only other comprehensive and open tool for this kind of analysis is LUPE [11]. However, we have not been able to find a validation of this tool compared with a clinical one. Furthermore, LUPE has been coded in the IDL programming language, which will force authors who wish to implement their own methods to acquire an IDL development license.

In this study we present the implementation of an open source DSC quantification tool developed as an ImageJ [22] plugin using the

* Corresponding authors at: Instituto de Investigación Sanitaria Gregorio Marañón, Madrid, Spain. Tel.: +34 91 426 50 67.
E-mail addresses: pmacias@hggm.es (P.M. Gordaliza), juanjose.vaquero@uc3m.es (J.J. Vaquero).
[1] Authors have contributed equally to this work.

http://dx.doi.org/10.1016/j.compbiomed.2015.01.002
0010-4825/

Java programming language and validated against Philips IntelliSpace, a widely used clinical tool. Our tool has been developed in a modular way to allow external researchers easily include their own quantification or preprocessing algorithms.

2. Materials and methods

2.1. Theoretical basis and mathematical description

The mathematical approach behind the quantification process has been extensively reviewed in the literature [10,17,16,21] and here we will provide a brief summary of the basic concepts.

For each voxel in the image, the signal drop after the contrast injection depends on the tracer concentration and can be modeled as

$$S(t) = S_0 \times e^{-(C_m(t)/k)} \tag{1.1}$$

where $S(t)$ is the change over time of the image signal for any given voxel, S_0 is the baseline signal before the contrast bolus arrival, $C_m(t)$ is the measured concentration of gadolinium as a function of time and k is a constant that depends on the scanner used to acquire the image series and on the TE of the acquisition sequence. Since the parameter k appears simultaneously in the numerator and denominator in the equations that compute the parametric maps, it cancels out and a value of $k=1$ is used on the next equations for simplification purposes.

From the previous formula, the expression for the contrast concentration can be obtained:

$$C_m(t) = -\ln\frac{S(t)}{S_0} \tag{1.2}$$

The shape of this concentration curve is heavily influenced by the way the tracer bolus is injected into the patient. To achieve an accurate quantification it is necessary to eliminate this effect from the concentration curve. The arterial input function (AIF) describes the way the tracer bolus reaches the main vessels; therefore, the concentration in a region can be expressed mathematically as the convolution of the AIF with an idealized contrast bolus ($C(t)$), as follows:

$$C_m(t) = C(t) \otimes \mathrm{AIF}(t) \tag{1.3}$$

The AIF is obtained from the image data via manual delineation, typically from the carotid arteries, if they are present in the field of view. Also, there are available robust algorithms to select automatically the relevant AIF voxels and avoid the manual delineation process [21]. Once the input function has been obtained, it is possible to compute the idealized contrast bolus for each voxel using deconvolution techniques, such as the one published by [10]. Please note that we follow the notation from [10], commonly used in nuclear medicine, but in other contexts $C(t)$ is referred to as $R(t)$ [27].

With all these curves computed, three parametric maps of interest, CBV, MTT and CBF, can be calculated. The expression for CBV is [10]:

$$\mathrm{CBV} = \frac{\kappa_H}{\rho} \times \frac{\int C_m(t)}{\int \mathrm{AIF}(t)} \tag{1.4}$$

In the last equation, the constant κ_H corrects for the different hematocrit between large and small vessels and has a value of 0.73, and ρ is the density of brain tissue (1.04 g/ml) [23].

The MTT is defined as

$$\mathrm{MTT} = \frac{\int C(t)}{C_{\max}} \tag{1.5}$$

where $C_{\max}$ is the maximum of $C(t)$, the contrast measurement after deconvolution with the AIF, for that voxel.

Finally, the CBF parametric map is obtained by dividing the previous ones:

$$\mathrm{CBF} = \frac{\mathrm{CBV}}{\mathrm{MTT}} \tag{1.6}$$

While these steps describe the basic process, there are some preprocessing steps that can be applied in order to reduce the influence of noise or undesired effects such as tracer recirculation or leakage through the blood–brain-barrier. These effects can also be considered in the model: it is a common practice to simply remove them by fitting each contrast curve to a gamma function that takes into account only the first pass of the tracer and assumes no leakage [19,5,28]. This function is defined by the following equation:

$$C_m(t) = K(t-t_0)^{\alpha} e^{-\frac{t-t_0}{\beta}} \tag{1.7}$$

for any moment $t > t_0$, where t_0 is defined as the contrast injection time. Once the contrast concentration curve for each voxel has been fitted to this function, the fitted data are used for the rest of the quantification process.

2.2. Software implementation

This work presents an open DSC quantification tool. The development platform chosen, ImageJ [22], is an imaging analysis and processing tool created by the National Institutes of Health (Bethesda, Maryland, USA) in the Java programming language (Oracle Corporation, Santa Clara, California, USA). ImageJ source code is available under a public domain license, which allows developers to implement new algorithms easily with the help of its well-documented application programming interface (API). This also allows concentrating on algorithm implementation, as the common imaging handling and processing functions (opening and saving image files, displaying them on screen, different basic filtering approaches…) are already implemented. For some mathematical computations as SVD (Singular Value Decomposition) or linear regression, we have used the Apache Commons Math libraries (http://commons.apache.org/proper/commons-math/).

The implementation of the DSC quantification process has been made in a modular way. This allows modifying the quantification workflow so as to easily replace or include new steps (for instance, new preprocessing algorithms or fitting models).

The processing workflow, from the moment the image has been loaded into ImageJ, is the following:

1. The image is masked to eliminate from the parametric computation all the voxels outside the subject body area. This masking process is done using a simple thresholding method.
2. The voxels from which the AIF should be computed are automatically detected using the algorithm detailed in [21]. This algorithm searches for those voxels that present contrast concentration curves with an earlier peak value, higher maximum amplitudes and smaller full-width half-maximum (FHWM). Intuitively, it tries to detect those voxels that carry contrast before it interacts with any of the tissues present in the image. Once the appropriate voxels have been selected, they are presented to the user as an overlay to the original image. If the automatic selection is not satisfactory to the user, it is possible to go back and choose a different AIF calculation mode. The present program version includes two other methods: manual delineation, in which the AIF is computed by averaging selected voxels, and importing the AIF from a text file. Other automatic or semi-automatic AIF calculation methods could be easily integrated.