Flexible Motion Models for GPU-accelerated MRI Simulations with Complex Motion and Flow using KomaMRI

Pablo Villacorta-Aylagas[1], Carlos Castillo-Passi[2], Rosa María Menchón-Lara [1], Joaquín Anatol-Hernández[3]
Pablo Irarrazaval[2], José Benito Sierra-Pallares[3], Carlos Alberola-López[1],
[1] Laboratorio de Procesado de Imagen, Universidad de Valladolid, Valladolid, Spain
[2] Department of Electrical Engineering, Pontificia Universidad Católica de Chile
[3] Dpto. de Ingeniería Energética y Fluidomecánica, Universidad de Valladolid, Valladolid, Spain

Summary (total: /300 characters):

KomaMRI.jl was extended to include (1) mix-and-match simple motions and (2) complex arbitrary motions, while maintaining high performance and GUI interactivity. These motion models are incorporated into the Phantom type, and an HDF5-based file format is provided to store and share Phantom objects.

Abstract (total: /5000 characters):

Introduction:
The integration of realistic motion into digital simulators to solve the Bloch equations has been previously proposed. For this, two approaches have been taken: optimizing performance,[1] or the inclusion of complex –including flow– motion.[2-4] Nevertheless, extensible GPU-accelerated simulators, like KomaMRI.jl,[5] could reconcile these two seemingly contradictory goals, as Koma is written in Julia, which facilitates fast and composable code. Currently, its phantom implementation defines motion by means of analytic functions, so simulation of realistic complex motion is limited, and the definition of even simple movements requires an analytic model. We propose an extension of the phantom structure to ease the definition of simple motion and to give room to complex arbitrary motion.

Methods:
The phantom structure currently existing in the simulator has been extended by adding a new *MotionModel* abstract type. Each motion model requires a method definition of the *get_spin_coords* function, which returns the positions of each spin at a set of arbitrary time instants –i.e. the time steps of the simulation–. Two models have been defined, namely:

1.- SimpleMotion allows the user to easily create simple movements by means of a few parameters. Internally, these parameters are used by *get_spin_coords* to define motion functions that apply to all the spins of the phantom (see Figure 1). So far, we have implemented translation, 3D-rotation, and heartbeat motion types, which can be both combined, as well as periodically extended. Translation is characterized by the linear displacements in the three axes, 3D-rotation by the rotation angles with respect to x (pitch), y (roll) and z (yaw), and heartbeat by the three known types of strain in cardiac motion, namely, circumferential, radial, and longitudinal.
Non-periodic types require the start and end of the movement, while periodic types require the period and a time asymmetry factor.

2.- ArbitraryMotion allows individual motion definitions for every spin, stored in matrices dx, dy and dz. Each row corresponds to a spin and each column to a discrete time instant. For periodic motion, the parameter *period_durations* stores the cycle duration. Pseudo-periodic motion –such as arrhythmias– is accounted for by letting this parameter be a vector with different durations. To include flow dynamics, we have defined an additional flag matrix –referred to as *resetmag*– which indicates the time instant at which a reset in the magnetization of a spin should be carried out. This is necessary for simulating inflow and outflow of spins. Figure 2A illustrates the *ArbitraryMotion* structure arrays.
In this case, the function *get_spin_coords* performs a piecewise linear interpolation, considering the dx, dy and dz values as nodes, as shown in figure 2B.

Phantom's including these motions can be stored and shared by using our proposed HDF5-based file format (.phantom).

Results:
Figure 3 shows a cine acquisition with tagging pre-pulses on a phantom with a combination of periodic heartbeat and rotation *SimpleMotion* types. For *ArbitraryMotion*, figure 4A illustrates six phases of three cine acquisitions. First and second rows correspond to a longitudinal and an axial slice of a cylindrical phantom respectively, in which blood particles are injected into the left-hand side of the cylinder and move rightwards. The bright-blood effect encountered in 2D GRE sequences can be observed in Figure 4B. This effect occurs due to fresh blood particles entering the imaging area where the steady state has been reached by the spins of the wall tissue. Finally, the third row shows a cine image executed on an XCAT[6] model where trajectories are expressed by a sequence of positions.

Discussion:

Although KomaMRI originally included a built-in method for defining motion using analytic functions, our approach makes it easier for the user to create simple motions by requiring only the definition of a few parameters. More involved motions require another model, such as the one we proposed (*ArbitraryMotion*). The advantage of our model with respect to other simulators[1-4] is that we have no difference in processing according to the type of motion we intend to simulate, since organ motion and flow –and, eventually diffusion– are handled similarly with minor changes in the simulation functions, due to Julia's multiple dispatch. This contribution has also led to the definition of a new *.phantom* file format, which follows the HDF5 standard and allows to store, share and visualize Phantom's easily. In terms of performance, *SimpleMotion* is as performant as the previous motion implementation in Koma. For *ArbitraryMotion*, there is a trade-off between pre-calculating positions, with higher memory consumption, and on-the-fly calculations, which increase simulation time.

Conclusion:
KomaMRI has been extended to accommodate simple and arbitrary motion models and a new file format has been defined for reproducibility. Research is now focused on *ArbitraryMotion* performance optimization.

References:
1. Xanthis C.G, Venetis I.E, Aletras A.H. High performance MRI simulations of motion on multi-GPU systems. J Cardiovasc Magn Reason. 2014; 16: 48.
2. Hanson HM, Eiben B, McClelland JR, van Herk M, Rowland BC. Technical Note: Four-dimensional deformable digital phantom for MRI sequence development. Med. Phys. 2021; 48: 5406–5413
3. Fortin A, Salmon S, Baruthio J, Delbany M, Durand E. Flow MRI simulation in complex 3D geometries: Application to the cerebral venous network. Magn Reson Med. 2018 Oct;80(4):1655-1665
4. Weine J, McGrath C, Dirix P, Buoso S, Kozerke S. CMRsim–A python package for cardiovascular MR simulations incorporating complex motion and flow. Magn Reson Med. 2024; 1-17.
5. Castillo-Passi C, Coronado R, Varela-Mattatall G, Alberola-López C, Botnar R, Irarrazaval P. KomaMRI.jl: An open-source framework for general MRI simulations with GPU acceleration. Magn Reson Med. 2023; 90: 329-342.
6. Segars W.P, Sturgeon G, Mendonca S, Grimes J, Tsui B.M.W. 4D XCAT phantom for multimodality imaging research. Medical Physics. 2010; 37: 4902-4915

Acknowledgements: The authors acknowledge grants PID2020-115339RB-I00, TED2021-130090B-I00 and PID2021-124407NB-I00 from the Ministerio de Ciencia e Innovación of Spain.

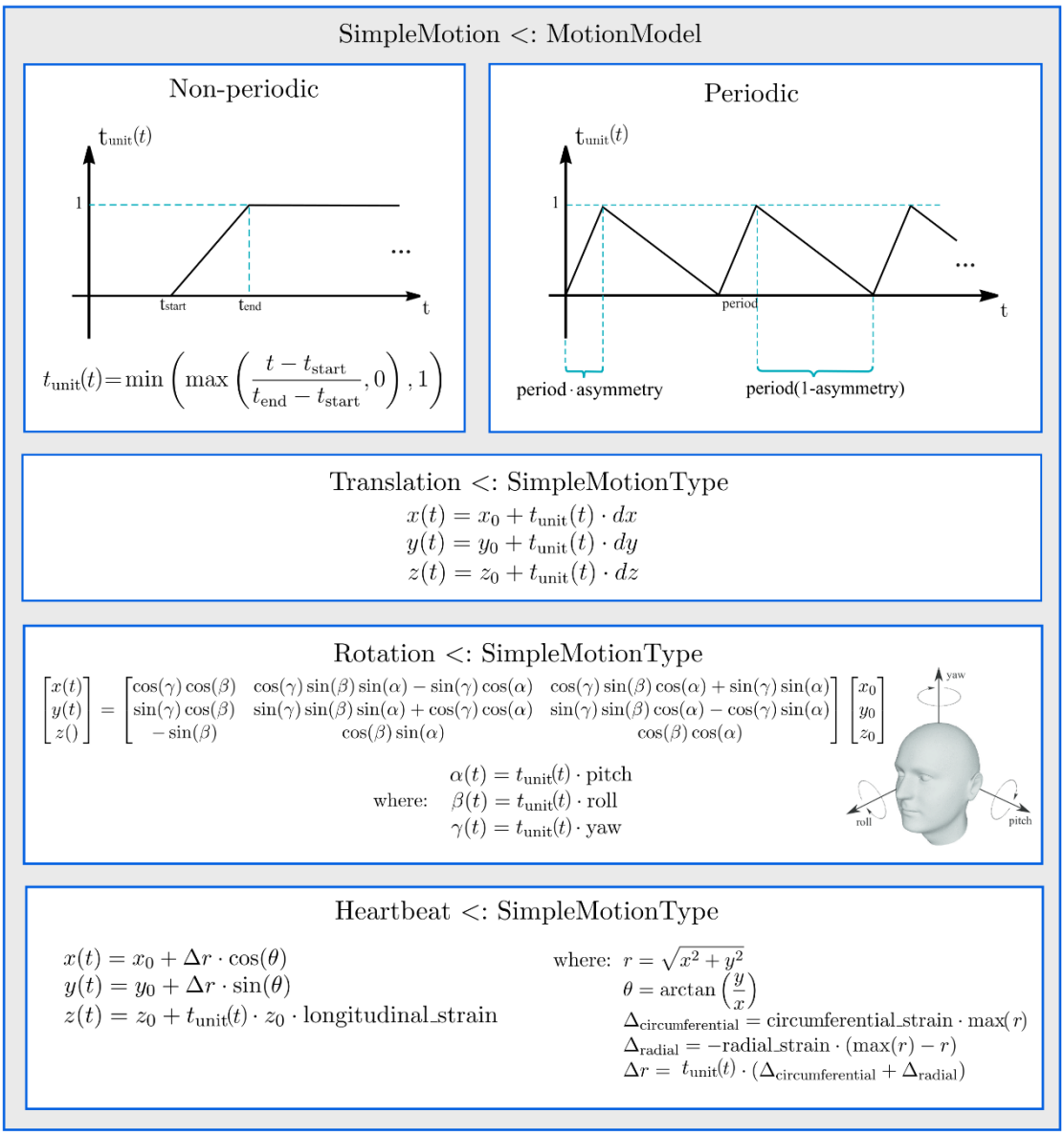

Figure 1. Implementation scheme of *get_spin_coords* function for *SimpleMotion* model. The way in which the variable t is transformed depends on the type of movement –periodic or non-periodic–. The transformation is carried out by the function t_unit. Spin positions are calculated using t_unit and the parameters of the simple motion type.

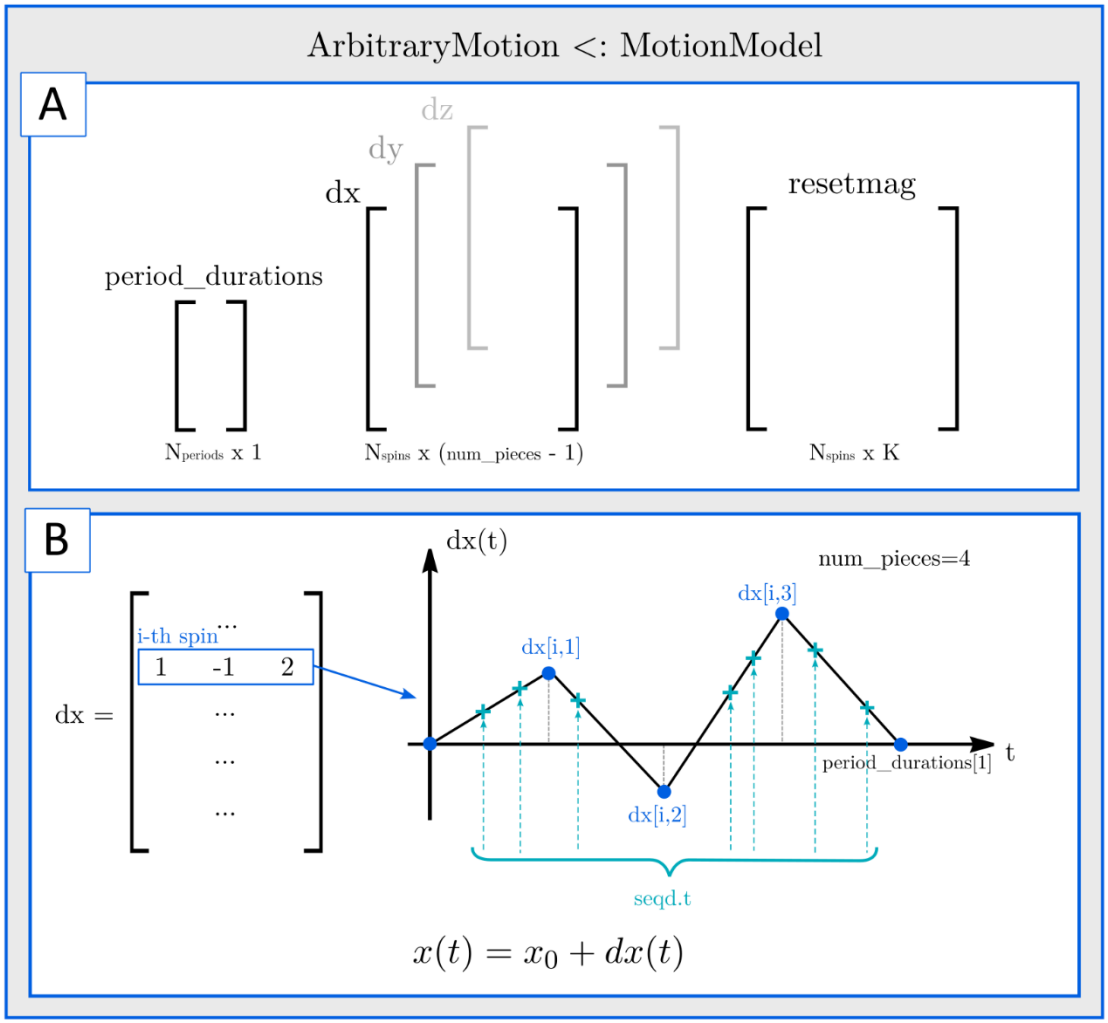

Figure 2. A) *ArbitraryMotion* model structure arrays. B) Implementation scheme of *get_spin_coords* for *ArbitraryMotion* model. For simplicity, only dx for the *i*-th spin is shown. Spin positions are calculated by means of a piecewise linear interpolation. The variable seqd.t represents the discretized sequence time points in which the motion is interpolated. This motion is assumed to be periodic, and therefore it starts and ends with zero displacement (those samples are not included in dx).

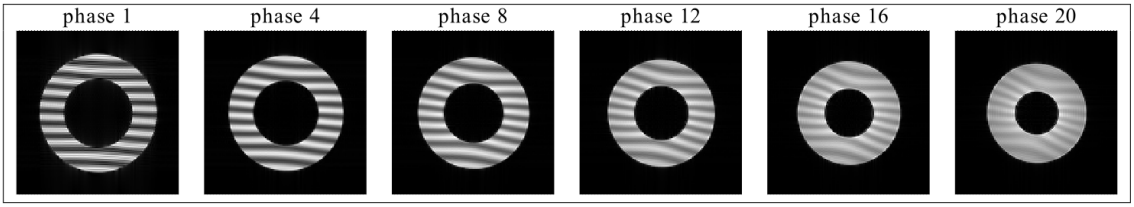

Figure 3. 2D cine tagging acquisition over a myocardium model phantom, composed by 128778 spins and presenting heartbeat and rotation movements at 60 bpm. 128 x 128 matrix, FOV = 12 cm. The sequence used is a bSSFP with TR = 30 ms and flip angle = 15º. Tagging is implemented by means of a simple SPAMM (1-1) sequence. Figure inspired in [1]. The tag lines fade due to T1 recovery.

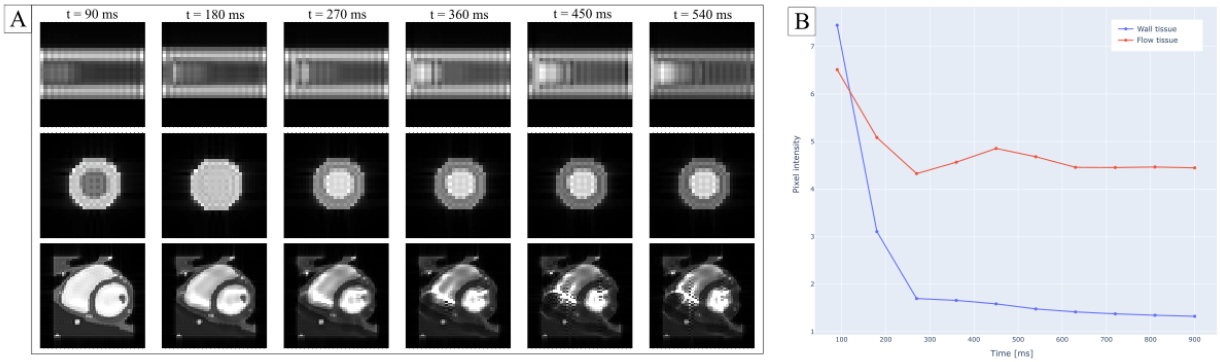

Figure 4. A) 2D CINE acquisitions over a cylindrical flow phantom and a XCAT heart phantom. Displacement fields obtained from XCAT were not fully realistic, resulting in some artifacts due to blood not being incompressible (giving unrealistically high proton densities). B) Graphical representation of the bright-blood effect over time for two pixels.