



King's Research Portal

Document Version
Peer reviewed version

[Link to publication record in King's Research Portal](#)

Citation for published version (APA):

E. Ovtchinnikov, D. Atkinson, O. Bertolli, C. O. da Costa-Luis, N. Efthimiou, R. Fowler, C. Kolbitsch, J. Matthews, C. Prieto, A. J. Reader, B. A. Thomas, C. Tsoumpas, M. Turner and K. Thielemans (2017). SIRF: Synergistic Image Reconstruction Framework. In *6th Conference on PET-MRI and SPECT-MRI (PSMR) 2017*

Citing this paper

Please note that where the full-text provided on King's Research Portal is the Author Accepted Manuscript or Post-Print version this may differ from the final Published version. If citing, it is advised that you check and use the publisher's definitive version for pagination, volume/issue, and date of publication details. And where the final published version is provided on the Research Portal, if citing you are again advised to check the publisher's website for any subsequent corrections.

General rights

Copyright and moral rights for the publications made accessible in the Research Portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognize and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the Research Portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the Research Portal

Take down policy

If you believe that this document breaches copyright please contact librarypure@kcl.ac.uk providing details, and we will remove access to the work immediately and investigate your claim.

SIRF: Synergistic Image Reconstruction Framework

Evgueni Ovtchinnikov¹, David Atkinson², Ottavia Bertolli³, Casper da Costa-Luis⁴, Nikos Efthimiou^{3,5}, Ronald Fowler¹, Christoph Kolbitsch⁶, Julian Matthews⁷, Claudia Prieto⁴, Andrew Reader⁴, Benjamin A. Thomas³, Charalampos Tsoumpas⁸, Martin Turner¹, Kris Thielemans³,

On behalf of *CCP PETMR*

¹ Visual Analytics and Imaging Systems Group, Science and Technology Facilities Council, UK; ² Centre for Medical Imaging of University College London, UK; ³ Institute of Nuclear Medicine of University College London, UK; ⁴ Biomedical Engineering Department of King's College London, UK; ⁵ School of Biological, Biomedical and Environmental Sciences, University of Hull, UK; ⁶ Physikalisch-Technische Bundesanstalt (PTB), Berlin, Germany; ⁷ Division of Informatics, Imaging and Data Sciences, MAHSC University of Manchester, UK; ⁸ Division of Biomedical Imaging of the University of Leeds, UK. Corresponding author: k.thielemans@ucl.ac.uk

Abstract— The combination of positron emission tomography (PET) with magnetic resonance (MR) imaging opens the way to more accurate diagnosis and improved patient management. At present, the data acquired by PET and MR scanners is processed essentially separately, and the search for ways to improve accuracy of the tomographic reconstruction via synergy of the two imaging techniques is an active area of research.

The aim of the collaborative computational project on PET and MR (CCP-PETMR), supported by the UK engineering and physical sciences research council (EPSRC), is to accelerate the research in synergistic PET-MR image reconstruction by providing an open access software platform for efficient implementation and validation of novel reconstruction algorithms.

In this paper we present a status update on the Synergistic Image Reconstruction Framework (SIRF) software suite from the CCP-PETMR.

Index Terms— Positron Emission Tomography, Magnetic Resonance Imaging, Software, Scientific Programming

I. INTRODUCTION

MR images usually provide excellent anatomic depiction and high soft tissue contrast of the investigated area, but may lack sensitivity for imaging probe detection compared to PET images. The latter in turn can clearly show regions of abnormality thanks to the distribution of radiotracer uptake, whereas normally functioning areas may not be seen at all. Recognition of the complementarity of the two imaging modalities prompted the development of integrated clinical PET-MR scanners, which as yet has not been paralleled by the development of an integrated software platform for processing data acquired by such scanners.

CCP-PETMR (www.ccpetmr.ac.uk) is a network founded in April 2015 with the aim of facilitating the investigation of novel synergistic PET-MR image reconstruction methods, by providing the PET-MR research community with a software development platform that would be simple enough to use for research and education purposes and, at the same time, powerful enough to be able to handle (in reasonable time) raw medical data acquired by PET, MR and PET-MR scanners. In this software both available integrated clinical PET-MR systems are being targeted, and a suitable data format exchange is being negotiated with the manufacturers. Support for PET and MR data is mostly complete for the Siemens mMR whilst for the GE Signa PET/MR only the PET data are currently supported.

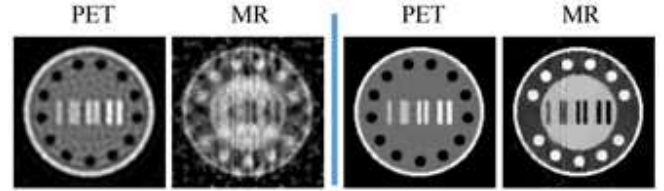


Fig. 1. Example reconstruction from simulated PET-MR data (Poisson noise for PET, Gaussian noise for MR, radial sampling with 15 lines). Left: independent reconstructions. Right: joint-reconstructions using Quadratic Parallel Level sets prior (after fig 5 in [1]).

II. SOFTWARE DEVELOPMENT STRATEGY

We strive to make our platform useful for as wide a research community as possible. An important choice is the programming language that the users of our platform will need to adopt for their own software development. Two languages that are widely used by researchers with limited software development experience are Matlab and Python, and we expect our users to have basic knowledge of at least one of them. To handle the conflict between the simplicity of development and the efficiency of the developed software we use advanced programming languages such as C++ for the actual development and provide simple Python and Matlab interfaces to building blocks that the users of our platform may need in their own software development. For ease of interfacing into Matlab, Python and prospective languages, we wrap C++ code into a C interface.

III. CURRENT SOFTWARE STRUCTURE

Our platform, the synergistic image reconstruction framework (SIRF), is built upon existing open source software. At present, we are employing two well-known libraries: STIR (software for tomographic image reconstruction) [2] for PET and Gadgetron [3] for MR, and we refer to particular reconstruction software packages as 'engines'. Current software structure is reflected by the SIRF file structure, which is as follows:

```
SIRF          : root
data          : raw data
examples      : engine-independent demos
src           : sources
iUtilities    : common C-interface utilities
xGadgetron    : Gadgetron extensions
  examples    : Gadgetron-specific demos
cGadgetron    : C/C++ interface to Gadgetron
mGadgetron    : Matlab interface to cGadgetron
```

pGadgetron : Python interface to cGadgetron
 xSTIR : STIR extensions
 examples : STIR-specific demos
 cSTIR : C/C++ interface to STIR
 mSTIR : Matlab interface to cSTIR
 pSTIR : Python interface to cSTIR

IV. USAGE

SIRF follows object-oriented principles, which means that most functions callable by a user's code are member functions (or methods) of a platform's objects.

As an example, a Python script using ordered subsets maximum *a posteriori* one step late (OSMAPOS) algorithm for PET reconstruction would involve the following lines (among others, not shown for brevity):

```
# create reconstructor object
recon = OSMAPOSReconstruction()
# set reconstructor parameters
recon.set_num_subsets(num_subsets)
# prepare reconstructor for use
recon.set_up(image)
# perform reconstruction
recon.reconstruct(image)
```

The above exemplifies the simplest level of user's involvement in PET reconstruction. While it is the most performance-efficient way of using the STIR library for reconstruction, some users may prefer to have more control over computation. For example, given an existing iterative algorithm, they may like to run a loop containing the following lines:

```
# perform one iteration
recon.update_current_estimate()
# get current image as ndarray
iarr=recon.get_current_estimate().as_array()
# process image using own processor
iarr=my_image_processor(iarr)
# make processed image a new image estimate
recon.set_current_estimate(image.fill(iarr))
```

On a still higher level of involvement, the user may employ their own or a third party optimization algorithm. For instance a simple gradient ascent could use a loop

```
# compute the gradient of objective function
grad=obj_fun.gradient(image,subset)
# get gradient as ndarray
g=grad.as_array()
# get current image as ndarray
x=image.as_array()
# define a line search function
f=lambda t: -obj_fun.value(image.fill(x+t*g))
# maximize f
t=scipy.optimize.fminbound(f, 0, maxstep)
```

Similarly for MR, the following simple example does not require any knowledge about the engine implementation.

```
input_data=AcquisitionData(input_file)
# remove oversampling
preprocessed_data = preprocess_acquisitions(input_data)
# apply your own preprocessing
my_preprocessed_data = my_preprocessor(preprocessed_data)
# create reconstruction chain
recon = SimpleReconstruction()
# reconstruct
image = recon.reconstruct(my_preprocessed_data)
```

The more knowledgeable user might want to speed this up by using Gadgetron-specific code by “chaining” 2 gadgets:

```
# get access to input data file
input_data=AcquisitionData(input_file)
# create Gadgetron chain that de-oversamples and
# reconstructs
recon=ImagesReconstructor(['RemoveROOversamplingGadget',
                           'SimpleReconGadgetSet'])
# reconstruct
image=recon.reconstruct(input_data)
```

This is the most performance-efficient way of using Gadgetron, however, some users may prefer to try their own acquisition preprocessor in addition to the removal of oversampling, in which case the above chain can be split into two, with user's preprocessing in between:

```
input_data=AcquisitionData(input_file)
# create chain that removes oversampling
acq_proc=
  AcquisitionsProcessor(['RemoveROOversamplingGadget'])
# remove oversampling
preprocessed_data = acq_proc.process(input_data)
# apply your own preprocessing
my_preprocessed_data = my_preprocessor(preprocessed_data)
# create reconstruction chain
recon = ImagesReconstructor(['SimpleReconGadgetSet'])
# reconstruct
image = recon.reconstruct(my_preprocessed_data)
```

The previous two examples are Gadgetron-specific and assume that the user is familiar with Gadgetron streaming architecture.

V. INSTALLATION

SIRF is available (under the Apache 2.0 license) at <https://github.com/CCPPETMR>. The user has the following options for obtaining and installing SIRF:

1. Download a Ubuntu Virtual Machine with all the necessary software except Matlab preinstalled from <https://www.ccppetmr.ac.uk/downloads>.
2. Linux/MacOS: Download the source of SIRF, STIR and Gadgetron and build (instructions on the [SIRF Wiki](#)).
3. Windows: as above but use pre-compiled Gadgetron from the VM (see the wiki for detailed instructions).

Conclusion and Future Work

This software suite will offer a new and simpler tool for new or established researchers to actively and faster participate in the research of PET and MR image reconstruction and be members of a larger consortium which should lead to faster translation of novel ideas in the clinical practice.

ACKNOWLEDGMENT

This work was supported by UK EPSRC under Grant EP/M022587/1.

REFERENCES

- [1] Ehrhardt, Thielemans et al. Joint reconstruction of PET-MRI by exploiting structural similarity, *Inverse Problems*, Vol 31, No 1, 2015
- [2] Thielemans, Tsoumpas et al., STIR: Software for Tomographic Image Reconstruction Release 2, *Physics in Medicine and Biology*, 57 (4), 2012 pp.867-883
- [3] Hansen MS, Sørensen TS. Gadgetron: An Open Source Framework for Medical Image Reconstruction. *Magn Reson Med*. 2013 Jun;69(6):1768-76.