DOI: 10.14529/jsfi160308

Data merging for the cultural heritage imaging based on Chebfun approach

Mariem El Afrit[1,2,3], *Yann Le Du*[1,2], *Rafaël Del Pino*[4], *Guolin Zhang*[5]

Cultural heritage imaging has specific needs with regards to the analysis of images that require the manipulation of a single digital object that combines the images obtained from different instruments probing different scales at different wavelengths, with the further possibility of selecting two or three dimensional representations. We propose a unified imaging data processing approach based on the "Chebyshev Technology" using the open source software Chebfun which, by mapping data processing to simple polynomial transformations, brought considerable improvements over already existing procedures. Within that same data processing framework we may further investigate how to merge images originating from different acquisition devices since all images are expressed in the same basis (an approximate Chebfun polynomial basis) before being merged. In the end, we hope to map all imaging data processing to simple polynomial operations. Our massive data-sets required parallelizing some Chebfun functions on GPUs, allowing about 100 times faster polynomial evaluation and up to 12 times faster on CPUs when parallelizing the whole algorithm.

Introduction

The Institut de Recherche de Chimie Paris (IRCP[6]) and Centre de Recherche et de Restauration des Musées de France (C2RMF[7]) have set up a mixed research team, Physico Chimie des Matériaux Témoins de l'Histoire (PCMTH), in order to bring new solutions to the challenges facing the analysis, conservation and restoration of cultural heritage objects, and one of the team's ambitious research projects concerns the latter's digital images : how to best acquire, store, analyze and combine them. The C2RMF tools of the trade include optical and X-ray photography [16], and, thanks to the expertise brought by the team's IRCP component in *Electron Paramagnetic Resonance* (EPR) who pioneered the application of *EPR imaging* (EPR-I) to exobiology [14], has decided to use EPR-I in order to image specific chemical species which X-rays and other traditional techniques are unable to specifically target, like the different layers of carbon-related material found inside paintings.

However, this EPR-I information needs to be merged with the one gathered from other sources, be it X-rays or optical photography in order to provide a single object to which we may attach an interface for subsequent manipulations. This merging would be greatly simplified if we could unify the different pipelines that take raw data and transform them into images. At the moment, there are different pipelines (programs and associated algorithms) for each imaging technique, and even the data formats are different. This actually stands as a major global challenge : many domain specific techniques exist to solve particular problems in imaging, yet the final results obtained after applying each technique are difficult if not impossible to combine. The PCMTH team, in association with the C2RMF imaging team, is thus developing a generic approach which would allow a single pipeline to process all the different kinds of data, with a single unifying data structure and mathematical model founded on *Chebyshev technology* [9], as championed by the Oxford University Numerical Analysis Group through the development of *Chebfun* [13], an open-source software system for numerical computing with functions. The mathematical basis of Chebfun is piecewise polynomial interpolation and in this paper, we first describe the

[1]PSL Research University,Chimie ParisTech-CNRS, IRCP, UMR8247, Paris, France

[2]Centre de Recherche et de Restauration des Musées de France, C2RMF, Paris, France

[3]Université Pierre et Marie Curie, Paris, France

[4]Ecole Normale Supérieure-Paris, Paris, France

[5]Ecole Nationale Supérieure de Chimie de Paris, Paris, France

[6]The IRCP is the research branch of *Chimie ParisTech*, covering many domains of chemistry.

[7]The C2RMF is an institution of the Ministry of Culture devoted to the analysis and conservation of the cultural heritage of the French Museums.

role Chebfun plays in our unifying approach to image data processing[8], and more specifically how we managed to fit the key tomographic process of *backprojection* into the Chebfun paradigm ; we show that this approach provides a promising imaging data processing unification without having to pay a performance cost : it is a zero-cost abstraction[9]. Secondly, we also describe how we managed, by working both on the algorithmic and hardware aspect, to accelerate our Chebfun paradigm application : the execution time speed was scaled down by orders of magnitude compared to our original straightforward implementation on general purpose hardware.

1. The mathematical model of EPR imaging

EPR is a technique that basically finds the value of a magnetic field at which chemical species absorb oncoming microwave radiations, as we can see in figure 1 : this is the signature of the chemical species.

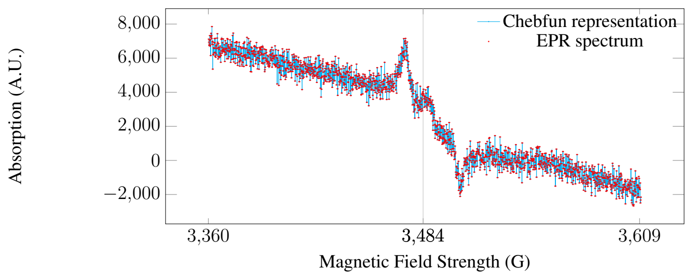


Figure 1. An EPR spectrum with its Chebfun representation. The absorption (arbitrary units) is measured as the magnetic field is varied (abscissa, in Gauss) and the microwave frequency of the oncoming radiation kept constant (around 9.5GHz for X band EPR); EPR traditionally measures the derivative of that signal, explaining the bumps and troughs. The maximum absorption is here around 3500 G

Now, the method of EPR imaging is similar to the well known one of Magnetic Resonance Imaging (MRI) : a magnetic field gradient maps the diffrent position of atoms sensitive to EPR to different positions on a spectrum, so if a species has a resonance spectrum s, then, given a magnetic gradient G and global magnetic field B, the linear density of the species is mapped to a spectrum in the following way :

$$r\left(B\right) = \int_{\text{sample}} c\left(x\right) \cdot s\left(B + G \cdot x\right) dx \tag{1}$$

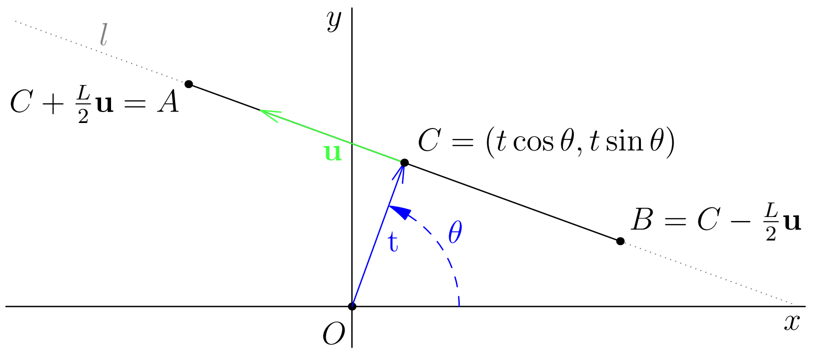
where the integral has the form of convolution, and is applied on the sample. The linear density c is itself related to the volume density ρ by a surface integral : $c(x)$ is the integral of ρ on the plane orthogonal to the direction of the magnetic field gradient at the position x on that direction. In order to simplify the problem, we shall consider our sample as being a 2D flat surface, thus allowing us to represent the surface integral as a line integral, yielding to the Radon transform $\mathcal{R}$ representation of c as being $c = \mathcal{R}\rho$, which allows us to rewrite equation 1 more abstractly as

$$r = s \star \mathcal{R}\rho \tag{2}$$

If we accept that simplification then the process of EPR imaging is depicted in figure (2).

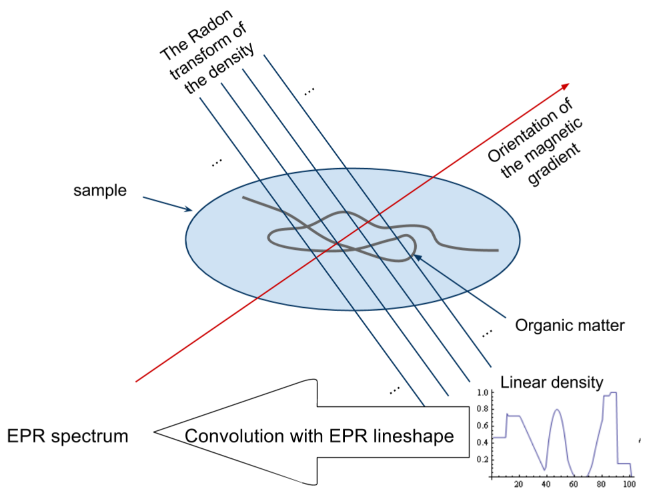
[8]We can find more details on our website HPU4science [2]

[9]We import that concept from the field of computer languages, especially from the abstractions provided by the C++ STL and implemented using Stepanov's generic programming approach [21].



y
l
$C + \frac{L}{2}\mathbf{u} = A$
$\mathbf{u}$
$C = (t\cos\theta, t\sin\theta)$
$B = C - \frac{L}{2}\mathbf{u}$
t
θ
O
x

Figure 2. Radon transform geometry : OC is the direction of the magnetic gradient, every spectrum is given by the choice of θ, and on a given spectrum, every data point has an abscissa t and for each t the ordinate on the spectrum is the integrated density along the direction BC. That geometry is used in our backprojection equations starting with 5, and underlies the more qualitative description given in figure 3



The Radon transform of the density
Orientation of the magnetic gradient
sample
Organic matter
Linear density
EPR spectrum
Convolution with EPR lineshape

Figure 3. An EPR spectrum is the Radon transform of the density of EPR sensitive chemical species computed for all (sampled) lines orthogonal to the magnetic field gradient. The information is blurred by a convolution with the absorption spectrum of the specific species under study, the *reference lineshape*, to which is added some unavoidable noise

The problem is now to find ρ, a flat surface density which thus depends on the variables (x, y), given r and s : it is a typical inverse problem, involving the well-known Fredholm integral equation of the first kind which the convolution equation 1 is an example of, together with the Radon transform, thus requiring two inversions, the second one being the backprojection operator $\mathcal{B}$. As for the convolutional part, this blurring of the density is traditionally dealt with using Fourier transforms directly on r, but it turns out that the deconvolution can be performed on the final image thanks to the linearity property of the backprojection and convolution operators :

$$\mathcal{B}\left(s \star \mathcal{R}\rho\right)(x, y) = \left(\mathcal{B}s \star \rho\right)(x, y) \tag{3}$$

This commutativity property allows us to postpone the deconvolution and apply it only on the blurred image, instead of applying it on the blurred linear density. But for this paper, we shall even further simplify the problem and suppose that the EPR spectra are a direct mapping of the linear density, a simplification fully justified by the property expressed in equation 3, and we shall thus simply inverse the simplified equation

$$r = \mathcal{R}\rho \tag{4}$$

but with the requirement that we want that inversion to be compatible with the Chebfun approach : we would like to find an inverse transform that maps to simple operations on the chebfun[10], without having to mediate those transformations through sampling.

2. The Chebfun compatible backprojection

In order to inverse the Radon transform, the common practice is to *filter* the backprojection, which we describe by the operator $\mathcal{B}$, and in the end we can retrieve the density ρ by using the well known Fourier form of the *filtered backprojection* [19]

$$\begin{gathered}\rho(x,y)\\=\\\tfrac{-1}{2}\mathcal{B}\left\{\mathcal{F}^{-1}\left[i\cdot\operatorname{sgn}(S)\,\mathcal{F}\left(\tfrac{\partial(\mathcal{R}\rho)(t,\theta)}{\partial t}\right)(S,\theta)\right](t,\theta)\right\}(x,y)\end{gathered}\tag{5}$$

which uses the parametrization described in figure 2. There exists an equivalent formulation that uses the *Hilbert transform* [19]

$$\rho(x,y)=-\frac{1}{2}\mathcal{B}\left[\mathcal{H}\left(\frac{\partial(\mathcal{R}\rho)(t,\theta)}{\partial t}\right)(t,\theta)\right](x,y)\tag{6}$$

which thanks to the commutativity of the Hilbert transform with the derivative becomes

$$\rho(x,y)=-\frac{1}{2}\mathcal{B}\left[\frac{\partial\mathcal{H}(\mathcal{R}\rho)(t,\theta)}{\partial t}(t,\theta)\right](x,y)\tag{7}$$

If we now recall that the $\mathcal{R}\rho$ are the EPR spectra, we can consider each of those to be a chebfun, which we shall call P_θ, and if we define

$$Q_\theta(t)=P_\theta(t)\cdot\sqrt{1-t^2}\tag{8}$$

we obtain that in equation (7),

$$\mathcal{H}(\mathcal{R}\rho)(t,\theta)=\mathcal{H}(P_\theta)(t)=\mathcal{H}\left(\frac{Q_\theta(t)}{\sqrt{1-t^2}}\right)(t)\tag{9}$$

we can now express Q_θ in the basis of the Chebyshev polynomials of the first kind

$$Q_\theta(t)=\sum_n\left(C_n(\theta)\cdot T_n(t)\right)\tag{10}$$

and from equations 9, and 10 we obtain

$$\mathcal{H}(P_\theta(t))(t,\theta)=\sum_n\left(C_n(\theta)\cdot\mathcal{H}\left(\frac{T_n(t)}{\sqrt{1-t^2}}\right)(t)\right)\tag{11}$$

Because we study physical objects which have a finite extension, and under the hypothesis that the corresponding chebfun will also be of finite support, our Hilbert transform becomes a *Tricomi transform* [22] $\mathcal{T}$, which allows us to rewrite [17] equation 11 as

$$\mathcal{H}(P_\theta(t))(t,\theta)=\sum_n\left(C_n(\theta)\cdot\mathcal{T}\left(\frac{T_n(t)}{\sqrt{1-t^2}}\right)(t)\right)\tag{12}$$

We now use a very useful property of the Tricomi transform, which is crucial in understanding how the filtered backprojection can indeed be mapped to simple operations on chebfuns :

[10]A *chebfun*, with a lower case "c" is a shorthand that means Chebfun object, where the uppercase "C" relates to the concept behind the Chebfun paradigm.

$$\mathcal{T}\left(\frac{T_n(t)}{\sqrt{1-t^2}}\right)(t) = -\frac{1}{n}\frac{\partial T_n(t)}{\partial t} \tag{13}$$

which allows us to rewrite equation 12 as

$$\mathcal{H}\left(P_\theta(t)\right)(t,\theta) = -\sum_n \left(\frac{C_n(\theta)}{n} \cdot \frac{\partial T_n(t)}{n \cdot \partial t}\right) \tag{14}$$

and finally equation 7 becomes

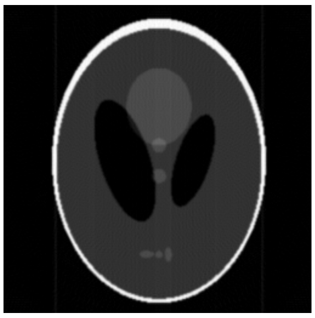
$$\rho(x,y) = \frac{1}{2}\mathcal{B}\left[\frac{\partial^2}{\partial t^2}\left(\sum_n \left(\frac{C_n(\theta)}{n} \cdot T_n(t)\right)\right)\right](x,y) \tag{15}$$

This last expression reveals the simple relationship that exists between the spectrum chebfuns, obtained directly from the raw (sampled) spectra, and the reconstructed image. It is quite straightforward to transform Equation 15 into an algorithm amenable to a Chebfun compatible implementation :

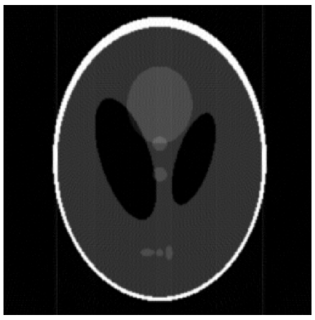
1. Prefactor the raw spectra $r(t)$ by $1/\sqrt{1-t^2}$;
2. transform each prefactored spectrum into a chebfun with the `FUNQUI` function ;
3. apply the square brackets part of equation 15 ;
4. apply the naive backprojection $\mathcal{B}$ that constitutes the remaining part of equation 15.

Part 4 of our algorithmic implementation above requires the computation of the naive backprojection, and here also we took advantage of the Chebfun approach.

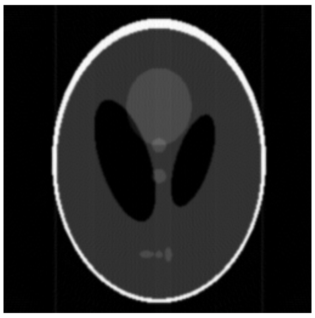
Figure 4 shows that our reconstruction is at least as good as the standard procedure using the vanilla Matlab imaging toolbox solution that uses the `IRADON` function.



Original Shepp-Logan phantom.



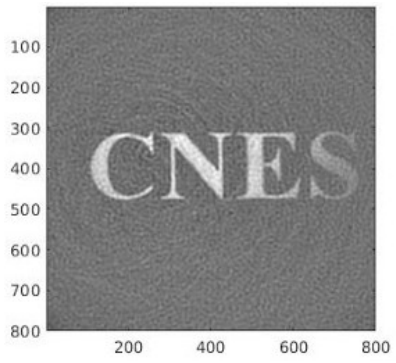
Standard Fourier reconstruction.



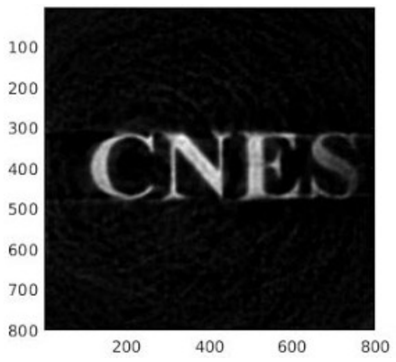
Our Chebfun reconstruction, cf. figure 6.

Figure 4. We tested our approach on the Shepp-Logan phantom (up). The other two figures show the phantom reconstruction using only the information provided by the Radon transform of the phantom sampled on the angles, cf. figure 2. We can see that our approach is at least as good as the standard one

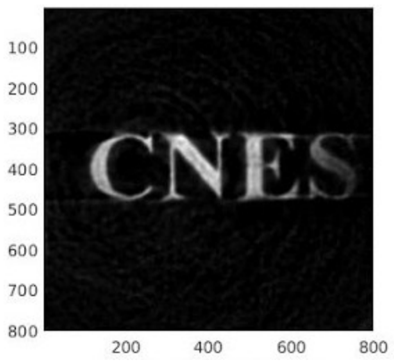
We have also applied our approach to real data [18], as we see in figure 5 : our approach does not yet procede to any noise filtering, yet the result is already an improvement on the traditional approach using a black-box Fourier approach as provided in the software suite that comes with the EPR imaging spectrometer [1].



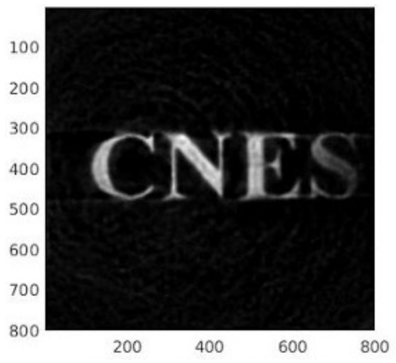
Standard Fourier reconstruction using the BRUKER Xepr software. It took few seconds for the construction but after a heavy manual work that took few hours.



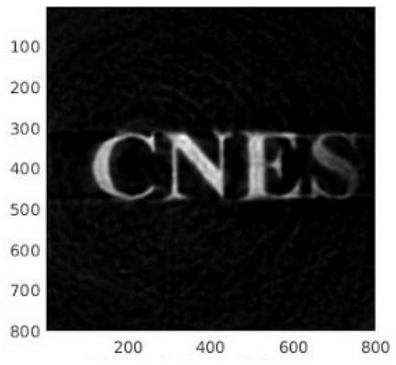
CPU sequential
Time : 6.6 hours



CPU parallel
Time : 33 minutes



GPU simple precision
Time : 2.5 minutes



GPU double precision
Time : 4 minutes

Our Chebfun reconstruction explained in figure 6.

Figure 5. "C(entre) N(ational) d' E(tudes) S(patiales)" (the French space institute) was laser printed (heated toner is very responsive to EPR) on a piece of paper (1cm by 5cm), hidden in an EPR tube filled with sand (EPR neutral) positioned in a Bruker EPR imaging spectrometer. The reconstructions do not (yet) include noise filtering

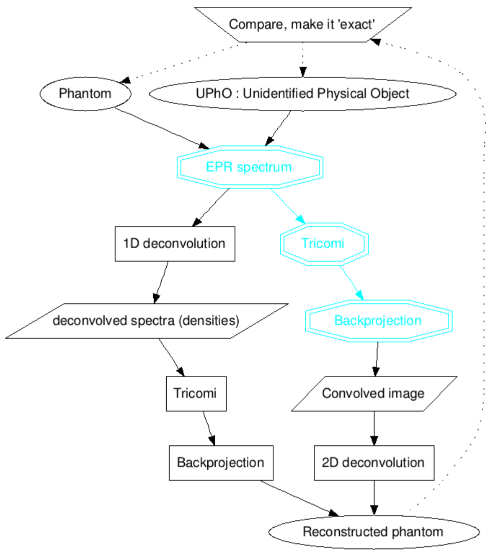


Figure 6. Our approach to tomography uses two mathematically equivalent yet implementation-wise different paths : in this paper, we describe the three nodes which are highlighted in blue and with double-line contours. The key steps are the transformation of spectra into chebfuns, and the formulation of the inverse problem solution using the Tricomi transform

3. Accelerating the code execution time

Because we target real time imaging, we need to decrease the processing time. The accelerations currently only rely on the vectorization of the algorithms, and in order to reap the full benefits we decided to buy Graphical Processing Units (GPU) hardware (Figure 7) and we describe that in sections 3.4 to 3.2. The implementation of the most important part of the code is described in section 3.1 and the whole code will be made available through a Gitlab repository as a literate program [20] written with the in-house developed Vim literate programming plugin[11] called *Kosmogram* [6]. Further improvements are expected with the use of a novel programming language, Pony [10], that puts actor programming to the forefront as we shall explain in section 3.3.

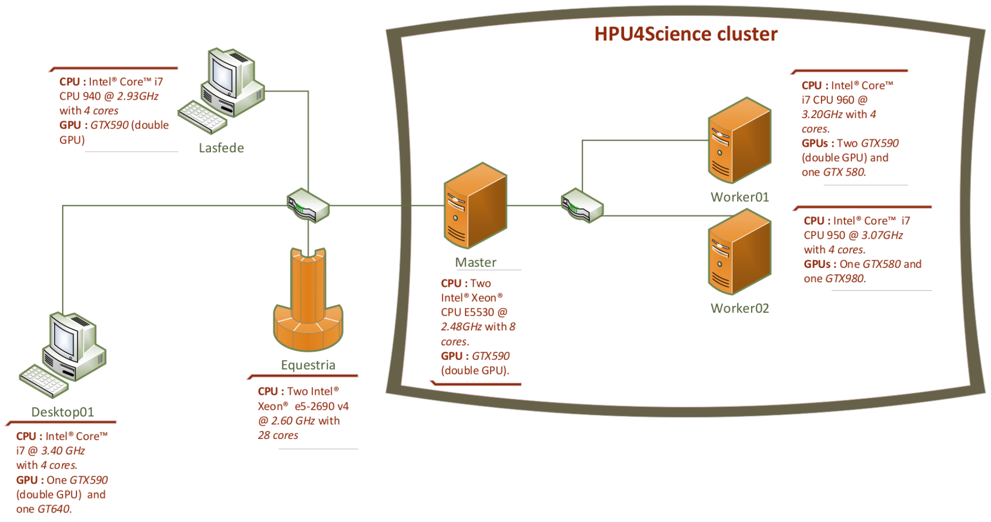

HPU4Science cluster
CPU : Intel® Core™ i7 CPU 940 @ 2.93GHz with 4 cores GPU : GTX590 (double GPU)
Lasfede
CPU : Intel® Core™ i7 CPU 960 @ 3.20GHz with 4 cores. GPUs : Two GTX590 (double GPU) and one GTX 580.
Worker01
Master
CPU : Two Intel® Xeon® CPU E5530 @ 2.48GHz with 8 cores. GPU : GTX590 (double GPU).
CPU : Intel® Core™ i7 CPU 950 @ 3.07GHz with 4 cores. GPUs : One GTX580 and one GTX980.
Worker02
Equestria
CPU : Two Intel® Xeon® e5-2690 v4 @ 2.60 GHz with 28 cores
Desktop01
CPU : Intel® Core™ i7 @ 3.40 GHz with 4 cores. GPU : One GTX590 (double GPU) and one GT640.

Figure 7. The cluster HPU4Science is composed of fours servers (Equestria, Master, Worker01, Worker02) and two desktops (Lasfede and Desktop01) with a total of 52 CPU cores and nine GPUs

[11]In section 3.3 we describe some more related open source project that we are working on.

3.1. Backprojection implementation

The key step in the imaging pipeline is the backprojection [19], which we express in the following form :

$$\mathcal{B}h\left(x,y\right) := \frac{1}{\pi}\int_{\theta=0}^{\pi} h\left(x\cdot\cos\left(\theta\right)+y\cdot\sin\left(\theta\right),\theta\right)\cdot d\theta \tag{16}$$

So we need to integrate on θ and we must therefore have the function that appears in the integrand, i.e. h as a function of $t = x\cdot\cos\left(\theta\right)+y\cdot\sin(\theta)$ which we have thanks to having transformed each spectrum into a chebfun : this means that for each θ_i we have a chebfun h_i which we compute at the value t, and we therefore need to construct the matrix of values of t. Once this is done, we obtain a matrix in which each line is the computation of a particular polynomial (chebfun) h_i on each t corresponding to a particular θ_i and all values of x and y in the image (Figure 8).

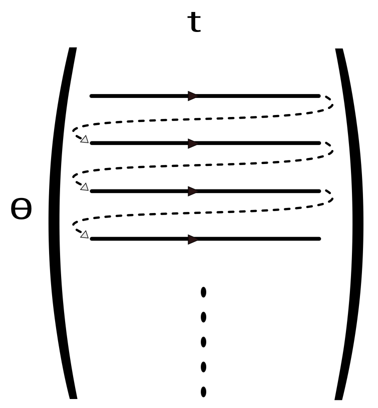


Figure 8. Construction of pixels corresponding to each t and θ

Therefore, if we look at the matrix column-wise (Figure 8), we have a set of values obtained for a particular t this time and all values of θ: we can now compute the chebfun for each column of values (by using `POLYFIT`), thus building a matrix of column chebfuns. Each chebfun is then summed to compute the backprojected value at all x and y: we sum (with the overloaded Chebfun sum) on each chebfun, we obtain a vector of size $x\cdot y$ which is the flattened image (Figure 9), and we can just `RESHAPE` it to an image.

We need to evaluate each θ polynomial on every $t\left(\theta,x,y\right) = x\cdot\cos\left(\theta\right)+y\cdot\sin\left(\theta\right)$ and in order to parallelize this task we construct a $N_\theta \mathrm{x} n^2$ matrix each row of which contains all the $t\left(\theta,x,y\right)$ for a given θ :

$\theta = \theta_1,\ldots,\theta_n$

$X = [x_1,\ldots,x_1,x_2,\ldots,x_2,\ldots,x_n,\ldots,x_n]$ each value appearing n times.

$Y = [y_1,\ldots,y_n,y_1,\ldots,y_n,\ldots,y_1,\ldots,y_n]$ this succession of values is repeated n times.

$t\left(i,:\right) = X\cdot\cos\left(\theta_i\right)+Y\cdot\sin\left(\theta_i\right)$.

For each line i of the matrix t we apply the polynomial h_i to obtain the matrix $t^{'}$:

Algorithm 1 Construction of the matrix $t^{'}$

```
for i=1:N_theta
        t(i,:) = h_i(t(i,:))
```

We then transform each column of the matrix t' into a chebfun objects and on each column we apply the function SUM, which is the integral of the chebfun :

Algorithm 2 Construction of the flattened picture

```
for i=1:n*n
        S(i) = sum(t(:,i))
```

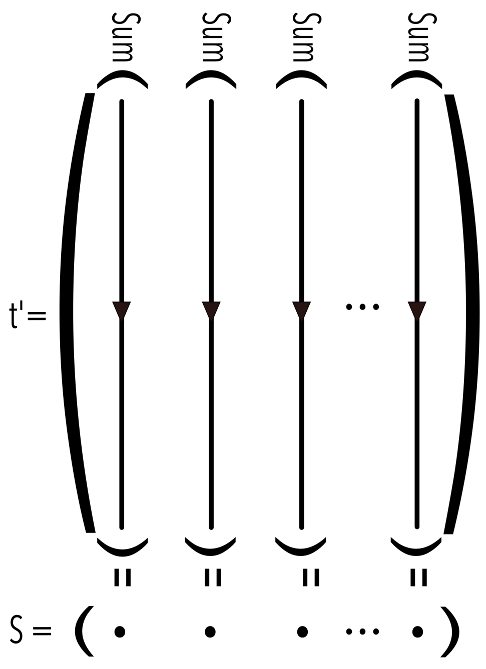


Figure 9. Flattened picture

In our code, wherever it was possible, we replaced the FOR loops by the function IRBSXFUN in the accelerated version.

3.2. MATLAB Distributed Computing Server

The image construction is done pixel by pixel, which is an embarrassingly parallel process which quiet naturally led us to use the MATLAB Distributed Computing Server [12] on the cluster HPU4Science which we designed accordingly as we shall see in section 3.4. The construction of the pixels is distributed on the appropriate target machines which we selected thanks to the MATLAB Job Scheduler (MJS) – Table 1 explains the MJS configuration for each construction. We then replace each FOR loop by a PARFOR loop.

Speedup results

In the beginning, the execution time took between 8 to 12 hours (depending on the sample under study) and thanks to the code vectorization (explained in subsection 3.1) we managed to significant-lygoing down to 6.6 hours on CPU. Then, using the CPU multi-core power, we further reduced the computation time to 33 minutes (about 14 times faster), and finally reached 4 minutes on GPUs in double precision (more than 100 times faster, or "orders of magnitude" as we claimed at the beginning of the paper). In table 1 we explain the MJS configuration used to get those results.

	MJS configuration	Time
CPU sequential	4 workers on Desktop01 (figure 7)	6.6 hours
CPU parallel	8 workers on the Master (figure 7	33 minutes
GPU simple precision	13 workers using 4 machines : two on Lasfede, two on Master, seven on worker05, two on Desktop01 (figure 7	2.5 minutes
GPU double precision	13 workers using 4 machines : two on Lasfede, two on Master, seven on worker05, two on Desktop01 (figure 7)	4 minutes

Table 1. The optimum configuration of resources used for each construction in Figure 5

3.3. Mixing parallelism and concurrency

Because we envision the application of this pipeline to data generated in real time – notably with the use of mobile imaging EPR probes – we have begun investigating the use of the actor paradigm to efficiently maximize CPU and GPU utilization, with the idea that as data gets collected, then we shall have as many actors constructing the output as there are increments in the data being collected: the first actor will construct a rough image from a small quantity of data, the second actor will improve on it by taking into account the first actor's data plus a newly collected one, and so on until the last actor generates the image from the whole collected data. That will allow users to decide when there is enough data collected for the task at hand, and, perhaps more importantly, decide if some modification to the imaging setup is needed, like changing the imaging probe position because the field of view is not adequate. The programming language Pony [10] [15] has really impressed us because it allows straightforward actor programming while at the same time staying on par with C with regards to speed. Our current focus in on developing a solid scientific library for Pony, by writing efficient wrappers for the GNU Scientific Library (GSL) [8], the ArrayFire [3] GPU optimized scientific library and the Yeppp SIMD library [11]. The development of these wrappers has taken the form of an open-source project called SciPony [7] on the Gitlab [5] platform, and is being developed using the literate programming Vim plugin called Kosmogram [6], itself also an open source project on Gitlab[12].

3.4. The HPU4Science cluster

The cluster, known as HPU4Science (Figure 7), began with a \$40,000 budget and a goal of building a viable scientific computation system with performance roughly equivalent to that of a \$400,000 "turnkey" system made by NVIDIA, Dell, or IBM.

The ideals of the project team required that the system use open source software wherever possible and that it be built only from hardware that is available to the average consumer. The project budget was, of course, an order of magnitude more than the average consumer could afford. In principle, however,

[12]Both SciPony [7] and Kosmogram [6] are open-source software, with the only requirement of having a Gitlab account.

anyone should be able assemble a similar, albeit scaled down, system and freely use the software and code developed by the HPU4Science team [4] to perform high-end scientific computing.

The HPU4Science cluster's construction went through multiple stages. We started by sharing an existing cluster room in Chimie-ParisTech (Figure 10 on the left) but because of electrical power constraints and heat control we had to move to another room that belongs to the PCMTH's team (Figure 10). We have recently (second half 2016) added new nodes in the cluster, with the goal of testing the actor programming framework as championed by the Pony programming language.

HPU4Science in the beginning

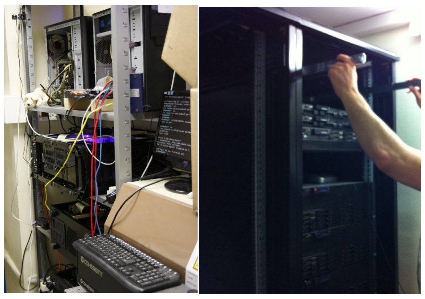

Servers room

Figure 10. The HPU4Science's current installation as of 2016 with 20 GPUs (4 GPU NVIDIA cards per node, for a total of 20 GPUs thanks to double GPU cards) with another node in the form of a 28-core blade (specifically for developing and testing a conconcurrent imaging pipeline) localized in a dedicated server room at Chimie-ParisTech

For the operating system we use Linux for its stability, the ability to pick from a wide variety of file systems, the large existing code base for high performance computing, and the ease of tailoring the OS to the specific hardware requirements. Availability of open-source projects with code that can be configured for highly parallelized processing was also an important advantage.

The master and all workers run Ubuntu server edition, installed with the minimal OpenSSH server profile. All machines are headless to minimize the OS memory footprint, so all interactions with the cluster happens at the command prompt through ssh.

The only closed source software used on the cluster is Matlab and it is because of the power of the Chebfun toolbox that we are re-coding some of its function using open source languages (3.3), as we explain in subsection 3.3.

4. Authors' contributions

Mariem El Afrit is a PhD candidate working on the development of novel approaches based on Chebfun for (tomographic) imaging and data fusion, and she worked specifically on the implementation of the algorithms and on the details of the mathematical derivations and their algorithmic formulation, together with the subsequent parallelization, while at the same time being responsible for most of the cluster assembling and management.

Yann Le Du is a CNRS research scientist at Chimie-ParisTech specialized in Electron Paramagnetic Resonance Imaging, and he initiated the use of Chebfun for imaging, devised the Tricomi transform approach to backprojection and worked on the whole mathematical framework, while also contributing to the code and launched the work on the re-coding of the pipeline in the Pony language.

Rafaël Del Pino, now a PhD candidate in cryptography, was a summer intern in the team under the guidance of Yann Le Du, and worked on the algorithm implementation, helped with the mathematical

derivations and specifically with the finalization of the mathematical form of the Tricomi expression used in the pipeline.

Guolin Zhang was a summer intern in the team, and worked on the acceleration of the code on the Matlab source code of the imaging pipeline under the guidance of Mariem El Afrit.

As for the paper itself, it was written by Mariem El Afrit and Yann Le Du.

Conclusion

Our goal is to unify imaging processing, especially tomographic imaging, by relying on a powerful data representation based on Chebfun [9]. This requires the adaptation of existing algorithms to this data structure, and we have succeeded in doing so with the most fundamental of the tomographic processes, the backprojection. By reformulating it in terms of the Tricomi transform, and applying the Cbebfun paradigm at each of the underlying algorithmic steps, we have managed to vectorize it and not only obtain results of the same quality as with the traditional approach, but with speeds that makes it possible to use our approach on real time data generation, and further work is thus underway to make the parallel processing concurrent[13].

References

1. *EPR User Manuals & Technical Documentation.*
2. HPU4Science website : http://hpu4science.org.
3. http://arrayfire.com.
4. http://hpu4science.org/overview/the-team.
5. https://gitlab.com/.
6. https://gitlab.com/hpu/codoc.
7. https://gitlab.com/hpu/scipony.
8. https://www.gnu.org/software/gsl/.
9. http://www.chebfun.org/about/.
10. http://www.ponylang.org/.
11. http://www.yeppp.info/.
12. *Mathworks. (2015). Global Optimization Toolbox: User's Guide (r2015b). Retrieved October, 2011 from http://fr.mathworks.com/help/mdce/index.html.*
13. Zachary Battels and Lloyd N Trefethen. An extension of MATLAB to continuous functions and operators. 25(5):1743–1770, 2003.
14. Laurent Binet, Didier Gourier, and Sylvie Derenne. Potential of EPR imaging to detect traces of primitive life in sedimentary rocks. *Earth and Planetary Science Letters*, 273:359–366, 2008.
15. Clebsch et al. Deny capabilities for safe, fast actors. 2015.

[13]We have applied for a funding to develop a mobile EPR spectrometer which requires real time imaging

16. Clotilde Boust et al. L'imagerie scientifique pour la conservation-restauration des oeuvres des musées : quels besoins en traitement d'images ? *GRETSI*, 2015.

17. Le Du et al. Electron Paramagnetic Resonance Imaging (EPR-I) with Chebfun , 2012.

18. Yann Le Du et al. The tricomi approach to chebfun imaging in electron paramagnetic resonance tomography : Towards a unifying imaging process in cultural heritage. *GRETSI*, 2015.

19. Timothy G. Feeman. The Mathematics of Medical Imaging. *The Mathematics of Medical Imaging*, pages 11–19, 2010.

20. Donald E. Knuth. Literate programming. *THE COMPUTER JOURNAL*.

21. Alexander A. Stepanov and Daniel E. Rose. *From Mathematics to Generic Programming*. Addison-Wesley Professional, 2014.

22. F. G. Tricomi. On the finite Hilbert transformation. *Quarterly Journal of Mathematics*, 2(March 1950):199–211, 1951.