TIGRE: a MATLAB-GPU toolbox for CBCT image reconstruction

2016 Biomed. Phys. Eng. Express 2 055010

(http://iopscience.iop.org/2057-1976/2/5/055010)

Biomedical Physics & Engineering Express

PAPER

TIGRE: a MATLAB-GPU toolbox for CBCT image reconstruction

Ander Biguri[1,3], **Manjit Dosanjh**[2], **Steven Hancock**[2] **and Manuchehr Soleimani**[1]

[1] Engineering Tomography Lab (ETL), Electronic and Electrical Engineering, University of Bath, Bath, UK
[2] CERN, Geneva, Switzerland
[3] Author to whom any correspondence should be addressed.

E-mail: a.biguri@bath.ac.uk **and** m.soleimani@bath.ac.uk

Keywords: cone beam CT, image reconstruction, tomography software, GPU

OPEN ACCESS

RECEIVED
3 April 2016

REVISED
29 June 2016

ACCEPTED FOR PUBLICATION
8 July 2016

PUBLISHED
8 September 2016

Abstract

In this article the Tomographic Iterative GPU-based Reconstruction (TIGRE) Toolbox, a MATLAB/CUDA toolbox for fast and accurate 3D x-ray image reconstruction, is presented. One of the key features is the implementation of a wide variety of iterative algorithms as well as FDK, including a range of algorithms in the SART family, the Krylov subspace family and a range of methods using total variation regularization. Additionally, the toolbox has GPU-accelerated projection and back projection using the latest techniques and it has a modular design that facilitates the implementation of new algorithms. We present an overview of the structure and techniques used in the creation of the toolbox, together with two usage examples. The TIGRE Toolbox is released under an open source licence, encouraging people to contribute.

1. Introduction

Among the techniques for x-ray computed tomography (CT) in widespread use, cone beam (CB) geometry is getting increasing attention nowadays, from medical imaging to material science. The possibility of reconstructing full 3D images using a reduced x-ray radiation dose is an important feature for CBCT development in medicine and it has led to high-quality 3D reconstruction in micro-CT [1]. Applications include maxillofacial imaging [2], guidance for radiation therapy in oncology [3], insect imaging [4] and material science [5].

In all applications of CBCT, the working principle is the same: 2D x-ray images of the 'sample' are obtained from different angles and a tomographic reconstruction algorithm is used to create an image from the data. The fact that in circular CBCT the original image is mathematically impossible to obtain [6, 7] and other factors, such as the high dimensionality of the problem or the inconsistency created by different physical effects with photons, make the image reconstruction problem what mathematicians define as *ill-posed*. Advanced mathematics is needed to generate a solution. This has led to extended research in image reconstruction algorithms, with a wide range of published approaches that give differing results. And it remains a hot topic.

While the use of CBCT is being increasingly extended to cover different imaging fields and research on reconstruction algorithms still sees new methods published, the end users of the images, both in medicine and microtomography, mainly use the simplest reconstruction algorithm, FDK [8]. This is worrying because, while FDK produces satisfactory images for good quality, full-projection, noiseless data, it performs poorly in less ideal scenarios. It has been repeatedly demonstrated that iterative algorithms [9] outperform FDK [10–14].

There are a few factors that influence the lack of connection between mathematics and usage. The main one is computation time. Most, if not all, alternative algorithms are iterative. They need to recompute repeatedly operations that are very memory- and computationally expensive, while FDK needs less time than a single such iteration. This is an important point especially in medical applications, where an image is needed rapidly as medical decisions are taken on the basis of what can be read from it. The time scales for iterative algorithms to run on a modern computer CPU are of the order of hours, days or even weeks for the most complex algorithms and bigger data. Another

factor is the lack of easy-to-use and free-distribution iterative algorithms. While some of the most recent toolboxes (presented later) do include some iterative algorithms, the vast majority of these algorithms have been completely ignored by both open source and commercial image reconstruction software. This lack, in conjunction with the fact that the research on reconstruction algorithms requires a deep understanding of such fields of mathematics as linear algebra and inverse problems, makes iterative algorithms somewhat out of reach for the end users of the reconstructed images.

In order to reduce the gap between algorithm research and end use, we have developed the Tomographic Iterative GPU-based Reconstruction (TIGRE) Toolbox, a MATLAB/GPU toolbox featuring a wide range of iterative algorithms. It uses the higher level abstraction of MATLAB with the lower (hardware-specific) performance of CUDA in order to make it fast and easy to use. In an attempt to bring the different fields together, we addressed the computation-time problem using the latest technologies in GPU computing with massive parallelization and memory management efficiency. Only the main computationally expensive blocks have been parallelized, with a modular design, allowing algorithm researchers easily to plug new methods together with the GPU blocks provided. Additionally, the algorithms can be used as single-line functions, giving total abstraction to researchers who are only interested in the resultant images, rather than in algorithm development.

Before explaining the specifics of the TIGRE toolbox, it is worth mentioning some other toolboxes that are also available. There are several commercial and free software packages for FDK reconstruction, including (but not exhaustively) CoBRA [15] , Ultrafast CB CT reconstruction [16] , OSCaR [17], Accelerating ConebeamCT [18]. Additionally, some more advanced toolboxes that include one or two iterative reconstruction algorithms (SIRT and/or CGLS) are also available, such as ASTRA [19], RTK [20] and 3D CB CT MATLAB [21]. Of these, ASTRA and RTK are the toolboxes that are most complete, however their infrastructure in low-level programming languages make them less suitable to work with when developing new algorithms.

In this paper we briefly describe the CBCT image reconstruction problem and some of the many ways to solve it. Thereafter, we give an overview of the structure of the TIGRE Toolbox and show some performance results and some reconstructed images. Finally, we discuss the future vision of the toolbox.

2. Methods

2.1. CBCT geometry

The geometry of CBCT can be represented as in figure 1. An x-ray source, S, is located at distance DSO from a centre of rotation O, where the origin of a cartesian coordinate system is located. The x-ray source irradiates a cone-shaped region containing the image volume $\mathbb{I}$ and a detector $\mathbb{D}$ measures the intensity of the photons impinging on it, photons that have been attenuated following the Beer-Lamber law. The image is centred at position O', which is displaced by $\overrightarrow{V_{\text{orig}}}$ from the coordinate system origin. The detector, located at distance DSD from the source and centred at D', has an offset of $\overrightarrow{V_{\text{det}}}$ from D, which is a point lying in the xy-plane at distance DSD − DSO from the origin. A projection coordinate system uv is defined centred at the lower left corner of the detector. During the measurement acquisition, the source and the detector rotate around the z-axis at an angle of α from their initial position.

The geometric variables described above are used in the TIGRE Toolbox to perform the necessary operations for image reconstruction, as shown in code snippet 1. It is worth mentioning that both $\overrightarrow{V_{\text{det}}}$ and $\overrightarrow{V_{\text{orig}}}$ are vectors that define a single offset per projection.

Code Snippet 1. Geometry definition in TIGRE.

```
%% Geometry structure definition.
% Distances
geo.DSD = 1536;                                 % Distance Source Detector
geo.DSO = 1000;                                 % Distance Source Origin
% Detector parameters
geo.nDetector = [512; 512];                     % number of pixels
geo.dDetector = [0.8; 0.8];                     % size in mm of each pixel
geo.sDetector = geo.nDetector.*geo.dDetector;   % total size of the detector in mm
% Image parameters
geo.nVoxel = [512;512;512];                     % number of voxels in the image
geo.sVoxel = [256;256;256];                     % total size of the image in mm
geo.dVoxel = geo.sVoxel./geo.nVoxel;            % size in mm of each voxel
% Offsets
geo.offOrigin = [0; 0; 0];                      % V_orig
geo.offDetector = [0; 0];                       % V_det
```

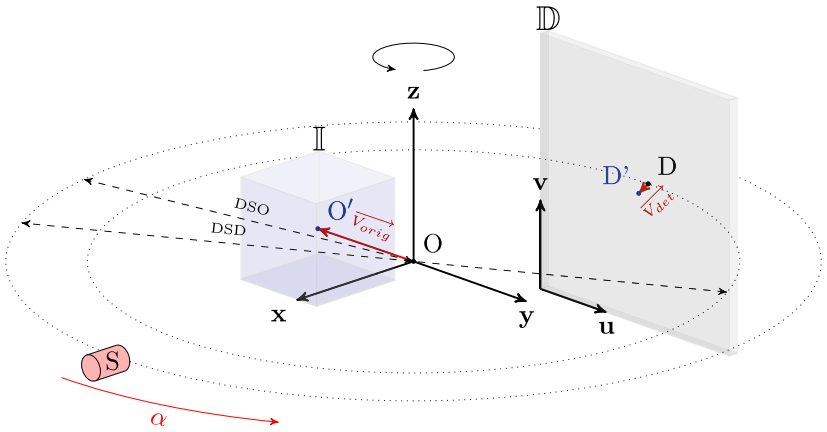

Figure 1. Geometry of CBCT.

2.2. Image reconstruction problem

For a given geometry, image reconstruction can be described by two different approaches. The first one is by solving the inverse Radon transform, a mathematical tool to describe the integral of a function over straight lines. The solution of the inverse Radon transform is well known in tomography and Feldkamp, Davis, and Kress modified it for CB geometries. Their solution is known as the FDK algorithm [22]. While the FDK algorithm will produce good quality images in an ideal scenario, it copes poorly with common unaccounted sources of error, such as beam hardening or photon scattering [23].

Alternatively, the image reconstruction problem has been described as a minimization one as in equation (1), where b are the projection data, x is the image and A is a matrix describing the intersections of x-rays and voxels in the image. In this equation, $G(x)$ is an optional term that describes a regularization functional $G(\cdot)$. This functional can be used to introduce additional constraints to the image reconstruction algorithm

$$\hat{x} = \text{argmin}_x \ \|b - Ax\|^2 + G(x). \qquad (1)$$

While the minimization formulation allows the use of advanced linear algebra techniques, there is a significant complication: the size of the matrix A. This is especially important as most, if not all, iterative techniques to solve equation (1) use one Ax and one A^Tb matrix-vector multiplication. As an example, matrix A for the geometry described in code snippet 1 for 360 projections has $94\,371\,840 \times 134\,217\,728$ elements with a sparsity index of 0.0017%. This requires 320 Gb of memory even using optimized sparse memory methods.

In order to cope with a problem of this scale, the most common approach is to substitute the matrix-vector multiplications Ax and A^Tb by operators $A(x)$ and $A^T(b)$, recomputing the relevant matrix values whenever necessary. While computationally very expensive to perform, the operators have an advantage: the values are completely independent of each other, making them suitable for parallel computing. In the TIGRE Toolbox, these two operators have been implemented using CUDA in a GPU capable of computing over 60K floating-point operations simultaneously[4]. This has resulted in speed-ups of up to 1400 times compared to matrix-based methods.

2.3. Toolbox structure

In this section an overview of the structure of the toolbox is given (see figure 2). As mentioned in the previous section, the main building blocks of any iterative algorithm are the so-called projection ($A(x)$) and back projection ($A^T(b)$) operators. In the TIGRE Toolbox, these two blocks have been optimized for GPU computing using CUDA. They lie in the lowest layer of the toolbox design and are constantly used by the other layers. The algorithms themselves lie in the topmost layer and are all coded in MATLAB, which provides the power and flexibility of a high-level language. To be able to communicate between the low-level, hardware-oriented CUDA and the high-level, design-oriented MATLAB, a set of the so-called *MEX functions* are needed. The toolbox has been designed not to have any specific data types or classes. Instead, it comprises only the basic MATLAB types, such as matrices and structures.

2.3.1. Projection and back projection

As already mentioned, the main building blocks of the toolbox are the CUDA/C++ implementations of the projection and back projection operators. Conceptually, the matrix A is a linearization of the model that describes the x-ray attenuation measured over a given domain and several different approaches to compute this may be found in the literature. Similarly, back projection is a 'smearing' over the domain with a weighting applied. Without explaining in detail all the

[4] In specific GPU models.

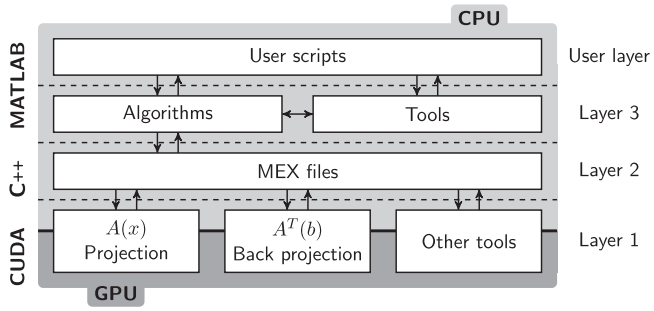

Figure 2. Structure of the TIGRE Toolbox.

methods available, we briefly describe those used in the toolbox.

For the projection, two approaches have been implemented: the voxel-ray intersection approach and the interpolation approach. The first of these uses the Siddon ray-tracing algorithm [24] with optimized operations [25]. This algorithm computes the distance between a given voxel and an infinitesimally narrow x-ray beam and multiplies that by the voxel intensity. This approach is the fastest way of computing the projector. However, it is known to introduce discretization square-block artefacts due to the finite size of the voxels, artefacts which become more significant the bigger the voxel size. To avoid this problem, a trilinear interpolation approach has been implemented where the path integral is evaluated every fixed Δl and image values are interpolated using advanced texture memory. To implement this an additional variable is added to the geometric definition of the problem: geo.accuracy $= 0.5$, which defines Δl as a fraction of the voxel size. This fraction is best chosen to be 0.5 or lower, as Jia *et al* [26] demonstrated.

For the back projection, two different approaches based on the same concept are used. Initially a ray is linked from the source location to the desired voxel, and extended to the detector. There, using bilinear interpolation, a value is read and added to that voxel with a weight. The difference between the two back projections is in this weight. One of them implements the FDK weight. However, this makes the back projection *unmatched*, i.e., it makes the back projection operator not equivalent to the transpose of the projector. While not important for most algorithms, this is crucial for Krylov subspace methods. In order to change that, a *matched* weight as described by Jia *et al* [27] is used. While not completely matched, they claim that it is above 99% similar to the transpose of matrix A. Both back projectors perform similarly.

2.4. Algorithms

One of the key features that we wish to introduce with the TIGRE Toolbox is algorithm variety. The field of image reconstruction has seen the development of a wide variety of methods to solve equation (1) using different solvers and regularization techniques. There are four main families of reconstruction algorithms present in the current implementation of the toolbox: the filtered back projection family, the SART-type family, the Krylov subspace method family and the total variation regularization family. A brief description of each algorithm subgroup follows, together with which algorithms are included in the toolbox.

The filtered back projection family is a set of algorithms based on solving the inverse of the Radon transform. Different variations of the algorithm have been proposed in the literature, but the toolbox contains just the standard FDK implementation with a small choice of filtering kernels[5].

The SART-type family [28] is set of algorithms that derives originally from the Kaczmarz method and is adapted to work projection by projection instead of row by row. This family of algorithms follows equation

$$x^{k+1} = x^k + \lambda^k V A^T W (b - Ax^k), \qquad (2)$$

where V and W are weight matrices based on ray length. The algorithms of this family mainly differ by the number of projections used simultaneously. In the TIGRE Toolbox, SIRT, OS-SART [29] and SART are implemented, where the image is updated using all projections, subsets of projections or projection by projection, respectively. Additionally, the toolbox provides different options for tuning the algorithms. For example different initialization techniques are implemented, such as FDK, multi-grid, or user-specified image. The main difference between the performance of the algorithms in this family is in convergence versus speed. The more data used in one update, the faster the algorithm will be per iteration, but slower (in number of iterations) to converge. For a more accurate solution, SART is suggested, while a faster result is obtained using SIRT, and with OS-SART somewhere in between.

Krylov subspace methods consitute a set of faster algorithms for solving linear equations. They iterate through Krylov subspaces, minimizing the eigenvectors of the residuals in descending order and so have

[5] FDK adapted from 3D CB CT MATLAB [21], with permission.

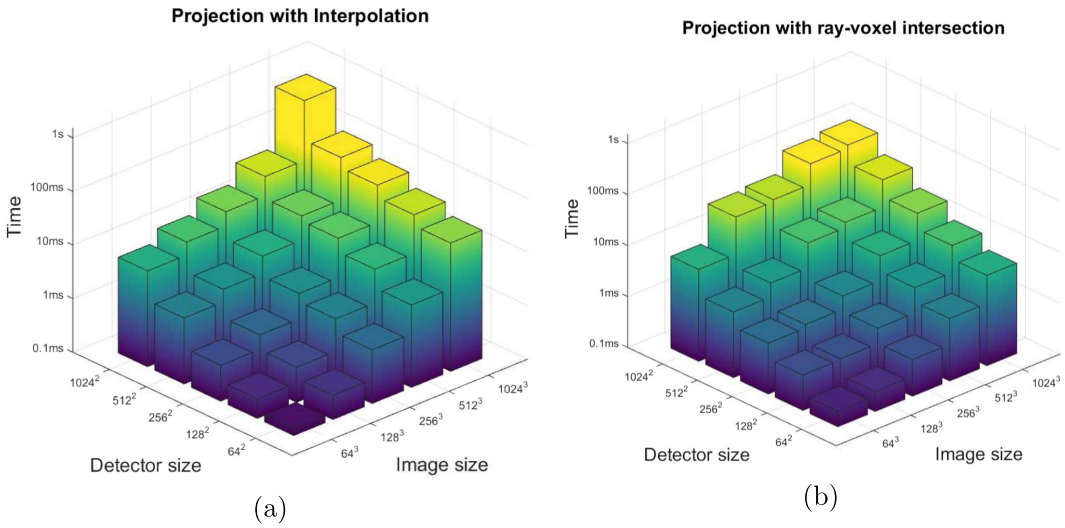

Figure 3. Projection operator performance (logarithmic scale) in GPU for a single projection and different image and detector sizes for interpolated mode (a) and ray-voxel intersection mode (b).

increased convergence rates compared to the SART family. From this family, the conjugate gradient least squares (CGLS) [30] has been added to the TIGRE Toolbox. This family of algorithms will get to a similar result compared to, for example SIRT, in approximately a tenth of the iterations while still having practically the same computational cost per iteration. These methods rely on iterating over the so-called 'Krylov subspaces', which are generated by the linear combination of the k first powers of A acting on b as in

$$K_r(A,\, b) = \text{span}\{b,\, Ab,\, A^2b, \ldots, A^{k-1}b\}. \qquad (3)$$

Finally, the total variation regularization family is included. The total variation norm is a common constraint in image denoising as it constrains the image to be piecewise smooth. It was introduced in CT with the advent of the ASD-POCS algorithm [31]. This set of algorithms is particularly good when the data are very noisy or the number of projections is reduced as the piecewise smoothness constraint forces the image into the least noisy state. From this family, the ASD-POCS (or POCS-TV), OSC-TV [32], B-POCS-TV-β [33] and SART-TV [34] (minimizing the Rudin–Osher–Fatemi model) are implemented. The total variation minimization has been partially GPU-accelerated. One of the limitations of this family of algorithms is that they require the tuning of more parameters than the other families, often needing to be tested several times until the optimal behaviour is found. The particularity of this family of algorithms resides in the double optimization of the problem. They minimize data first, using some of the algorithms from the previously mentioned families and after they minimize the 'total variation', essentially the noise in the image. The main difference between them is in the tools used in each of these minimization steps. ASD-POCS, OSC-TV and B-POCS-TV-β generally perform better with a limited amount of data, while SART-TV has an important role in reconstructing data from noisy projections. The minimization problem can be mathematically expressed as in

$$\hat{x} = \text{argmin}_x \; \|b - Ax\|^2 + \|x\|_{\text{TV}}. \qquad (4)$$

In addition to image reconstruction algorithms, some basic tools for image denoising, plotting, loading data and quality measurement are included in TIGRE. These include projection and image plotting utilities, an image denoising function, a CBCT cropping tool and a projection noise simulator among others. The toolbox contains demos to illustrate the usage of all algorithms, with extensive explanations of each of the parameters that they may require. There are also help pages for each of the functions that are included.

3. Results

In this section we demonstrate how the toolbox works by giving some performance figures and by showing some examples of image reconstructions. Before getting to the results, it is worth mentioning the specifications of the computer on which the toolbox has been developed and tested. The computer is a 64 bit Windows 7, with an Intel® Core™ i7-4930K 3.40 GHz CPU, and 32 Gb of RAM, running MATLAB® (R2014b, The Mathworks, Cambridge, UK) on a NVIDIA Tesla K40 GPU.

3.1. Performance

The performance of the GPU-accelerated projection and back projection is shown in figures 3 and 4,

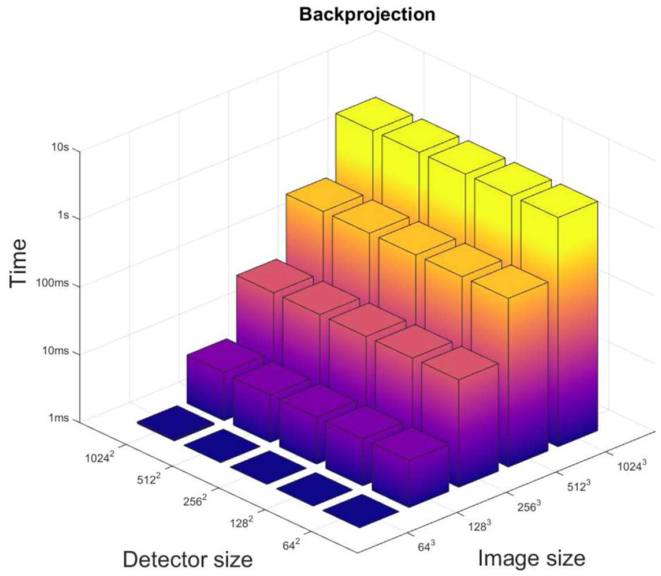

Figure 4. Back projection operator performance (logarithmic scale) in GPU.

respectively. In all these tests memory allocation time has been ignored.

The code has been run for a single projection with varying image and detector sizes for both projection types, interpolated and ray-voxel intersected. The resultant times go up to a second per projection for a 1024^3 image size and a 1024^2 detector size, which is considerably larger than a standard medical device. The interpolation test has been performed taking a sample every half voxel and, with this sampling rate, the ray-voxel intersection algorithm is about 4 times faster than the interpolation one. For the geometry in code snippet 1 (taken from a medical device), the ray-voxel intersection algorithm computes the projection every 10 ms.

A single back projection test is shown as both back projections perform similarly. Note also that, as expected, back projection performance is independent of the detector size.

3.2. Sample code

To illustrate the use and functionality of the toolbox we present two examples, one with phantom data and the other with data obtained from the RANDO head phantom at the Christie Hospital, Manchester, UK.

3.2.1. RANDO head reconstruction

We will first demonstrate the reconstruction of the RANDO head phantom using three different algorithms with the geometry defined in code snippet 1. The data set contains 360 equidistant projections. Once the data have been loaded using the code of snippet 2, the results of figure 5 can be obtained without the need for any more code. Information about total computation time and computation time per iteration are shown. Only some of the possible options are shown in the snippet, but more customization is possible. We refer the reader to the published documentation for advanced options and for insight into their numerical ranges.

Code Snippet 2. RANDO head data reconstruction.

```
% Define Geometry & load data
% From the data, the projection angles (in
% radians) must have been read
alpha = ...
%% Reconstruct image with different algorithms
% FDK
imgFDK = FDK(data,geo,alpha);
% CGLS
iterCGLS = 15;
imgCGLS = CGLS(data,geo,alpha,iterCGLS);
% OS-SART with multi-grid initialization
iterOSSART = 70;
imgOSSART = OS_SART(data,geo,alpha,iterOSSART,…
  'BlockSize',20,'Init','multigrid');
```

3.2.2. Reconstructions with few projections

The second test uses the 3D Shepp–Logan phantom to demonstrate the difference in image quality created by different algorithms in the case of few projections. Using just 20 projections, an image is reconstructed using FDK, OS-SART and ASD-POCS. The code snippet 3 demonstrates how to load data, set up parameters and reconstruct a limited amount of projection data. The results are shown in figure 6 and emphasize the distinct behaviour of these algorithms in certain scenarios.

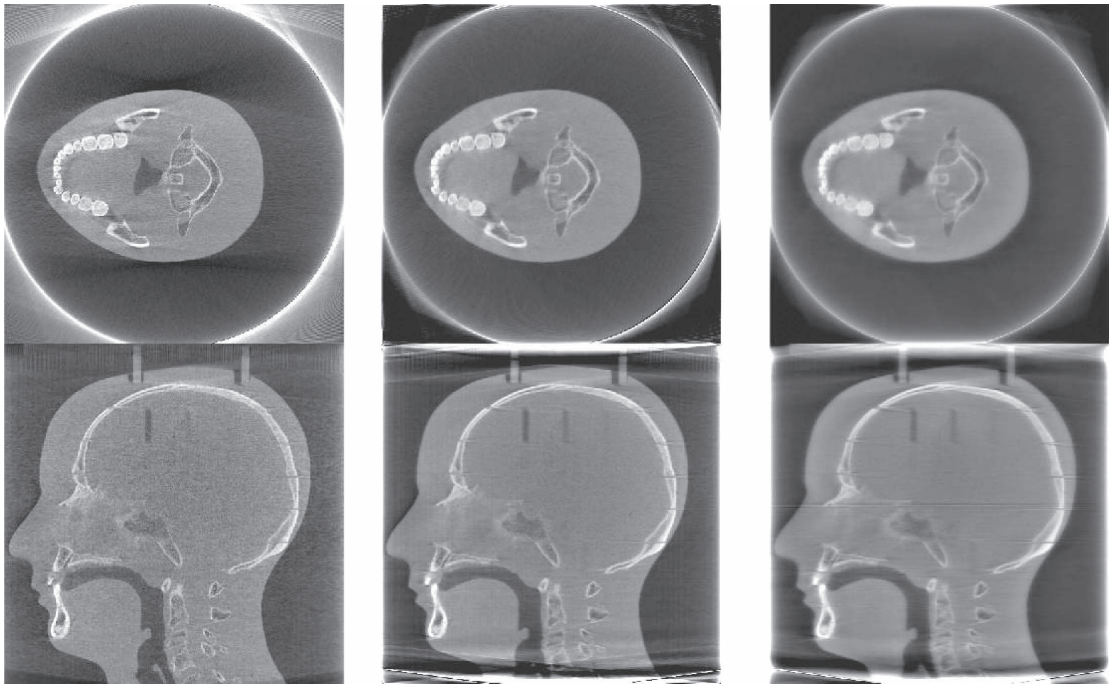

Figure 5. Cross-section of the image reconstructed from data from the RANDO head phantom by three different algorithms. FDK has noise across the entire image and significant strike artefacts (see top image). OS-SART and CGLS create a smoother image, removing most artefacts and having a clearer separation between tissues.

Code Snippet 3. Limited data reconstruction.

```
% Define Geometry
% Load phantom data
img = sheppLogan3D;
% Define angles
alpha = [0:18:360]*pi/180;
% Create projections
proj = Ax(img,geo,alpha);
%% Reconstruct images
imgFDK = FDK(proj,geo,alpha);
imgOSSART = OS_SART(proj,geo,alpha,50,…
  'BlockSize',5);
imgASDPOCS = ASD_POCS(proj,geo,alpha,50,12);
```

3.3. Implementation of an algorithm using TIGRE

To demonstrate the facility with which anyone can develop new algorithms using the TIGRE toolbox, we present in this section a side-by-side comparison of an algorithm definition and its TIGRE equivalent code, using the GPU accelerated features. For the sake of brevity, the CGLS algorithm has been chosen.

In table 1 the definition of the CGLS iterations and the implementation in TIGRE are shown. From the code snippet, we would like to highlight the limited use of library-related functions, as one of the strengths of TIGRE for the developer point of view is the easy-to-use application programming interface. The only difference in the code from a completely standard MATLAB script is the use of the function *Ax*() and *Atb*(), the main building blocks of the toolbox, as described in section 2.3. This allows anyone with MATLAB code for solving image reconstruction to easily modify their code by just changing the matrix-vector operations by TIGRE GPU functions.

Note that the functions inside TIGRE do generally have more code than the one shown here, as several options and performance enhancing MATLAB tools are used.

4. Discussion

In this paper we have presented a MATLAB/CUDA toolbox for fast 3D x-ray image reconstruction. While the toolbox has reasonably good performance—reducing to minutes an image reconstruction with complex iterative algorithms—and a wide variety of tools, improvements are possible.

The projection and back projection operators have been fully implemented in the GPU, but the algorithms are fully in CPU so a memory management overhead exists because the data need to be introduced and extracted from the GPU twice per iteration. This design has been proposed in order to have the

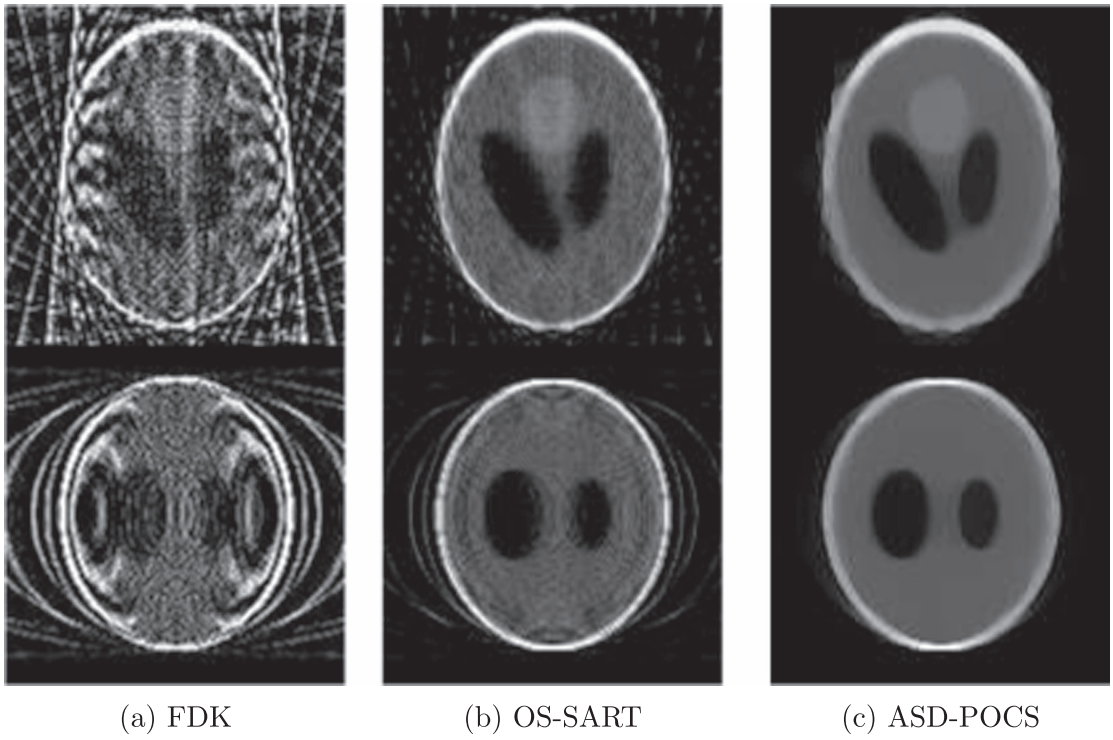

Figure 6. Cross-section of the image reconstructed from data from 20 projections of the 3D Shepp–Logan phantom. While an extreme case, the increased performance of the minimization algorithms over FDK is evident, especially for ASD-POCS. Times for this limited-projection example are all below a minute.

algorithms in a high-level language, as an algorithm implementation cycle in a low-level language like C++ is significantly longer than in MATLAB. We estimate that if the algorithms were written in C++/CUDA directly, an improvement in computation time of up to 50% could be achieved in some cases. However, this would increase the difficulty of adding new algorithms to the toolbox. We consider that the advantages of a high-level programming language for new algorithms are better than the possible benefits of doubling the speed, which is already reasonably good. Comparing the forward and back projection speeds to the ASTRA toolbox, TIGRE is 2.4 times slower. This can be easily explained by two factors. Firstly, the geometric options for CBCT are more flexible in TIGRE than in ASTRA, thus requiring more floating-point operations. Secondly, ASTRA implements an advanced ray splitting that increases memory latency in the GPU and that makes use of overlaps between x-ray paths at different angles [35]. However, due to the use of algorithms that do not compute adjacent angles together (such as SART or OS-SART), such an exploit has not been used in TIGRE, increasing the memory read time in GPUs. The same thing applies to back projection. Adding all the discussed effects that would decrease the time performance, all algorithms run about 5 times more slowly in TIGRE than in ASTRA, which constitutes the state of the art. Numerically, the differences between ASTRA and TIGRE are in absolute value of the order of 10^{-3}, which is about 0.01% in relative terms. This difference can be attributed to accumulated floating point errors due to different numerical approaches in the GPU code.

To speed up further the projection and back projection operators, a multi-GPU approach [36] could also be taken. Currently, TIGRE does not support multi-GPU architectures. A further weakness of the toolbox is the small number of functions for data loading and post-processing. However, we are presenting a first release and work will be continued, hopefully filling this gap in the near future. Another limitation of TIGRE comes from the current limitations in GPU technology. Currently, 12 GB is the maximum amount of memory on a GPU board, thus limiting the possible size of the images that can be reconstructed. Nevertheless, there is no problem to reconstruct a 1024^3 image with most algorithms so the maximum image size is still big.

The TIGRE Toolbox has been designed with the objective of reducing the gap between image reconstruction research and the end users of tomographic images. While research in reconstruction creates new algorithms every year, end users only have access to FDK implementations. With these two groups in mind, the toolbox:

- has easy-to-use 'black box' algorithms, making it extremely straightforward for researchers who are only interested in the quality of the images to test

Table 1. CGLS algorithm as definition, and implemented in TIGRE.

$x_0 = 0;\ d_0 = b;\ r_0 = A^T b;\ p_0 = r_0;$
$t_0 = Ar_0;\ \gamma_{k-1} = ||r_0||^2;$
for k = 1 **to** k = maxiter
  $\alpha_k = \gamma_{k-1}/||t_{k-1}||^2$
  $x_k = x_{k-1} + \alpha_k t_{k-1}$
  $d_k = d_{k-1} - \alpha_k t_{k-1}$
  $r_k = A^T d_k$
  $\gamma_k = ||r_k||^2$
  $\beta_k = \gamma_k/\gamma_{k-1}$
  $p_k = r_k + \beta_k p_{k-1}$
  $t_k = Ap_k$
end

```
%Initialize variables
x = zeros(geo.nVoxel');
d = b;
r = Atb(b,geo,angles,'matched'); %TIGRE
p = r;
t = Ax(r,geo,angles); %TIGRE
gamma_1 = norm(r(:));
%Loop until user defined maxiter
for k = 1:maxiter
  alpha = gamma_1/norm(t(:));
  x = x + alpha * t;
  d = d - alpha * t;
  r = Atb(d,geo,angles,'matched');%TIGRE
  gamma = norm(r(:));
  beta = gamma/gamma_1;
  gamma_1 = gamma;
  p = r + beta * p;
  t = Ax(p,geo,angles); %TIGRE
end
%x is the solution.
```

different algorithms without them requiring any knowledge of how the algorithms work;

- has easy-to-use building blocks (projection and back projection operators) that allow algorithm developers to test new methods using a high-level programming language but with the performance of the lowest level, GPU languages.

The code is released as open source, allowing anyone to download, test, modify and improve it. We enthusiastically encourage the submission of improvements, bug-fixes, demos, data and whatever else might help the community. Likewise, we encourage algorithm developers to submit their new algorithms to the toolbox, giving them visibility. Finally, we would like to encourage x-ray image end users to include data or descriptions of specific challenges they may have, allowing dialogue and hopefully leading to better ways of creating enhanced tomographic images.

While the toolbox was originally designed for CBCT image reconstruction, an option for 3D parallel-beam CT reconstruction has also been included allowing for more geometries, e.g., synchrotron data. Further tweaking the geometry structure of the toolbox would also permit 2D fan- and parallel-beam reconstructions.

The minimum requirements to run the toolbox are strongly dependent on the image size desired, as memory is the strongest limiting factor both on the CPU and GPU side. Generally speaking, any NVIDIA GPU with a compute capability higher than 3.5 would be sufficient to reconstruct arbitrarily large images. We recommend having at least 3 times the desired image size in GPU memory and 8 times in RAM in the computer. As an example, for a 512^3 image, 2 GB of GPU memory and 6 GB of computer RAM is the suggested minimum. The computing power (number of processors in the GPU and processor performance of the CPU) will have a strong effect on the speed of image reconstruction. Thus we recommend a state of the art CPU and a computing oriented GPU, such as from the Tesla family.

5. Conclusions

A 3D tomographic reconstruction toolbox has been developed with fast GPU-based algorithms and a wide variety of tools and image reconstruction algorithms. While TIGRE has been created for CBCT imaging, it can be used for any geometry, especially 3D geometries. With this toolbox we hope to make advanced algorithms more accessible to researchers and to provide a platform on which applied mathematicians and image users can work and collaborate. It will thus facilitate the comparison of such advanced algorithms with those in more common usage and so demonstrate the potential to achieve the same image quality with fewer projections and hence less dose. Future developments will include possible performance optimization, more algorithms, additional support for file formats and post-processing algorithms. With the TIGRE Toolbox, we hope to build a bridge between imaging communities and provide a platform where they can interact via software. The entire package is available at https://github.com/CERN/TIGRE.

Acknowledgments

The authors would like to acknowledge CERN Knowledge Transfer and EPSRC grant 1431573 for the funding, and NVIDIA for the hardware donation (Tesla k40c GPU) for research purposes that made it possible to achieve this work.

References

[1] Machin K and Webb S 1994 Cone-beam x-ray microtomography of small specimens *Phys. Med. Biol.* **39** 1639
[2] Vos W De, Jan Casselman and Swennen G R J 2009 Cone-beam computerized tomography (cbct) imaging of the oral and maxillofacial region: a systematic review of the literature *Int. J. Oral Maxillofacial Surg.* **38** 609–25

[3] Boda-Heggemann J, Lohr F, Wenz F, Flentje M and Guckenberger M 2011 kv cone-beam ct-based igrt *Strahlentherapie Und Onkologie* **187** 284–91

[4] Garwood R, Ross A, Sotty D, Chabard D, Charbonnier S, Sutton M and Withers P J 2012 Tomographic reconstruction of neopterous carboniferous insect nymphs *PLoS One* **7** 1–10

[5] Atwood R C, Jones J R, Lee P D and Hench L L 2004 Analysis of pore interconnectivity in bioactive glass foams using x-ray microtomography *Scr. Mater.* **51** 1029–33

[6] Tuy H A 1983 An inversion formula for cone-beam reconstruction *SIAM J. Appl. Math.* **43** 546–52

[7] Smith B D 1985 Image reconstruction from cone-beam projections: necessary and sufficient conditions and reconstruction methods *IEEE Trans. Med. Imaging* **4** 14–25

[8] Pan X, Sidky E Y and Vannier M 2009 Why do commercial ct scanners still employ traditional, filtered back-projection for image reconstruction? *Inverse Problems* **25** 123009

[9] Soleimani M and Pengpen T 2015 Introduction: a brief overview of iterative algorithms in x-ray computed tomography *Phil. Trans. R. Soc. London* A **373** 20140399

[10] Beister M, Kolditz D and Kalender W A 2012 Iterative reconstruction methods in x-ray CT *Phys. Med.* **28** 94–108

[11] Pontana F, Pagniez J, Flohr T, Faivre J, Duhamel A, Remy J and Remy-Jardin M 2010 Chest computed tomography using iterative reconstruction vs filtered back projection: I. Evaluation of image noise reduction in 32 patients *Eur. Radiol.* **21** 627–35

[12] Pontana F, Pagniez J, Flohr T, Faivre J, Duhamel A, Remy J and Remy-Jardin M 2010 Chest computed tomography using iterative reconstruction vs filtered back projection: II. Image quality of low-dose ct examinations in 80 patients *Eur. Radiol.* **21** 636–43

[13] Coban S B, Withers P J, Lionheart W R B and McDonald S A 2015 When do the iterative reconstruction methods become worth the effort? *13th International Meeting on Fully Three-Dimensional Image Reconstruction in Radiology and Nuclear Medicine (Fully3D 2015), 31 May - 04 June 2015, Newport, Rhode Island, USA*

[14] Yan H, Cervino L, Jia X and Jiang S B 2012 A comprehensive study on the relationship between the image quality and imaging dose in low-dose cone beam CT *Phys. Med. Biol.* **57** 2063

[15] Sheats M J and Stupin D M 1998 Cobra: cone beam computed tomography (CT) reconstruction code in interactive data language (idl) *Review of Progress in Quantitative Nondestructive Evaluation* (Berlin: Springer) pp 395–401

[16] Bronnikov A V Ultra-fast cone-beam CT reconstruction software http://bronnikov-algorithms.com/Products.htm

[17] Rezvani N *et al* OSCaR: open source cone-beam reconstructor (www.cs.toronto.edu/~nrezvani/OSCaR.html)

[18] Leeser M, Mukherjee S and Brock J 2014 Fast reconstruction of 3d volumes from 2d CT projection data with GPUs *BMC Res. Notes* **7** 582

[19] van Aarle W, Jan Palenstijn W, De Beenhouwer J, Altantzis T, Bals S, Batenburg K J and Sijbers J 2015 The {ASTRA} toolbox: A platform for advanced algorithm development in electron tomography *Ultramicroscopy* **157** 35–-47

[20] Rit S, Vila Oliva M, Brousmiche S, Labarbe R, Sarrut D and Sharp G C 2014 The reconstruction toolkit (RTK), an open-source cone-beam CT reconstruction toolkit based on the insight toolkit (ITK) *J. Phys.: Conf. Ser.* **489** 012079

[21] Kyungsang K 3D cone beam CT (CBCT) projection backprojection FDK, iterative reconstruction MATLAB examples https://mathworks.com/ matlabcentral/ fileexchange/35548 Version 1.14

[22] Feldkamp L A, Davis L C and Kress J W 1984 Practical cone-beam algorithm *J. Opt. Soc. Am.* A **1** 612–9

[23] Boas F E and Fleischmann D 2012 Ct artifacts: causes and reduction techniques *Imaging Med.* **4** 229–40

[24] Siddon R L 1985 Fast calculation of the exact radiological path for a three-dimensional ct array *Med. Phys.* **12** 252–5

[25] Han G, Liang Z and You J 1999 A fast ray-tracing technique for TCT and ECT studies *1999 IEEE Nuclear Science Symp. 1999. Conf. Record* vol 3, pp 1515–8

[26] Jia X, Yan H, Cerviño L, Folkerts M and Jiang S B A GPU tool for efficient, accurate, and realistic simulation of cone beam CT projections **12** 7368-78

[27] Jia X, Dong B, Lou Y and Jiang S B 2011 Gpu-based iterative cone-beam CT reconstruction using tight frame regularization *Phys. Med. Biol.* **56** 3787

[28] Andersen A H and Kak A C 1984 Simultaneous algebraic reconstruction technique (sart): a superior implementation of the art algorithm *Ultrason. Imaging* **6** 81–94

[29] Censor Y and Elfving T 2002 Block-iterative algorithms with diagonally scaled oblique projections for the linear feasibility problem *SIAM J. Matrix Anal. Appl.* **24** 40–58

[30] Qiu W, Titley-Peloquin D and Soleimani M 2012 Blockwise conjugate gradient methods for image reconstruction in volumetric CT *Comput. Methods Programs Biomed.* **108** 669–78

[31] Sidky E Y and Pan X 2008 Image reconstruction in circular cone-beam computed tomography by constrained, total-variation minimization *Phys. Med. Biol.* **53** 4777

[32] Matenine D, Goussard Y s and Després P 2015 GPU-accelerated regularized iterative reconstruction for few-view cone beam CT *Med. Phys.* **42** 1506

[33] Xue H, Zhang L, Cheng Z, Xing Y and Xiao Y 2010 An improved TV minimization algorithm for incomplete data problem in computer tomography *2010 IEEE Nuclear Science Symp. Conf. Record (NSS/MIC)* pp 2621–4

[34] Jia X, Lou Y, Lewis J, Li R, Gu X, Men C, Song W Y and Jiang S B 2011 GPU-based fast low-dose cone beam CT reconstruction via total variation *J. X-Ray Sci. Technol.* **19** 139–54

[35] Palenstijn W J, Batenburg K J and Sijbers J 2011 Performance improvements for iterative electron tomography reconstruction using graphics processing units (gpus) *J. Struct. Biol.* **176** 250–253

[36] Yan H, Wang X, Shi F, Bai T, Folkerts M, Cervino L, Jiang S B and Jia X 2014 Towards the clinical implementation of iterative low-dose cone-beam CT reconstruction in image-guided radiation therapy: cone/ring artifact correction and multiple GPU implementation *Med. Phys.* **41** 111912