Computer Methods and Programs in Biomedicine 218 (2022) 106725

Contents lists available at ScienceDirect

Computer Methods and Programs in Biomedicine

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

High-performance reconstruction of CT medical images by using out-of-core methods in GPU

Gregorio Quintana-Ortí[a,1], Mónica Chillarón[b,*], Vicente Vidal[c], Gumersindo Verdú[b,2]

[a] Depto. de Ingeniería y Ciencia de Computadores, Universitat Jaume I, Castellón, 12.071, Spain
[b] Instituto de Seguridad Industrial, Radiofísica y Medioambiental, Universitat Politècnica de València, Valencia, 46.022, Spain
[c] Depto. de Sistemas Informáticos y Computación, Universitat Politècnica de València, Valencia, 46.022, Spain

ARTICLE INFO

Article history:
Received 27 October 2021
Revised 18 February 2022
Accepted 28 February 2022

Keywords:
CT
QR factorization
Medical image
Reconstruction
Out-of-core
HPC
GPU

ABSTRACT

Background and objective: Since Computed Tomography (CT) is one of the most widely used medical imaging tests, it is essential to work on methods that reduce the radiation the patient is exposed to. Although there are several possible approaches to achieve this, we focus on reducing the exposure time through sparse sampling. With this approach, efficient algebraic methods are needed to be able to generate the images in real time, and since their computational cost is high, using high-performance computing is essential. *Methods:* In this paper we present a GPU (Graphics Processing Unit) software for solving the CT image reconstruction problem using the QR factorization performed with out-of-core (OOC) techniques. This implementation is optimized to reduce the data transfer times between disk, CPU, and GPU, as well as to overlap input/output operations and computations. *Results:* The experimental study shows that a block cache stored on main page-locked memory is more efficient than using a cache on GPU memory or mirroring it in both GPU and CPU memory. Compared to a CPU version, this implementation is up to 6.5 times faster, providing an improved image quality when compared to other reconstruction methods. *Conclusions:* The software developed is an optimized version of the QR factorization for GPU that allows the algebraic reconstruction of CT images with high quality and resolution, with a performance that can be compared with state-of-the-art methods used in clinical practice. This approach allows reducing the exposure time of the patient and thus the radiation dose.

1. Introduction

Computed tomography (CT) continues to be one of the most widely used medical imaging tests for the diagnosis and monitoring of patients. However, concerns about its safety have been growing in recent years. Since it uses ionizing radiation through X-rays, CT represents a risk for especially vulnerable patients such as pediatric patients, as well as recurrent patients who need to follow up on the evolution of a disease. Several studies [1–4] have shown the relationship between high exposure to X-rays and the possibility of developing different types of cancer such as leukemia, brain cancer, breast cancer, thyroid cancer, among others. Another study [5] claims that the maximum dose that does not pose a risk to the patient is around 100 milliSieverts, which is equivalent to approximately 10 abdominal studies in an adult. Exposure above this threshold for adults or within this range for pediatric patients can increase the risk of pathologies caused by ionizing radiation.

In consequence, research efforts in this area must focus on developing methods that reduce the dose required to perform a CT scan in order to minimize the risk for patients. There are different approaches to reducing the radiation dose. The two most common methods are based on the reduction of the tube current [6,7] and the reduction of tube voltage [8]. Both are usually known as "low-dose CT" and employ analytical methods, such as the Filtered Back-Projection (FBP) [9], since they have a very low computational cost to reconstruct the images. Although these methods with low-dose projections do not attain an optimal image quality, they are the basis of the more evolved iterative methods (IR), such as iDose (by Philips Healthcare) and SAFIRE (by Siemens Medical Solutions). These iterative techniques can outperform FBP by applying approximations and corrections on the images in each iteration, using the statistical information available from the scanner and from previ-

* Corresponding author.
E-mail addresses: gquintan@uji.es (G. Quintana-Ortí), mnichipr@inf.upv.es (M. Chillarón), vvidal@dsic.upv.es (V. Vidal), gverdu@iqn.upv.es (G. Verdú).
[1] ORCID ID: 0000-0002-7912-7826.
[2] ORCID ID: 0000-0001-5098-080X.

https://doi.org/10.1016/j.cmpb.2022.106725

ous reconstructions. Despite their higher computational cost, they are well established in clinical practice and most manufacturers use their own optimized iterative method to process low-dose projections.

On the other hand, the possibility of reducing the radiation dose by using a sparse sampling scheme has been gaining popularity recently. By using this scheme, the exposure time is reduced since the acquisition is not continuous and the number of projections is reduced, thus decreasing the radiation dose absorbed by the patient during a scan. Although there are no commercial scanners using this type of acquisition, there are prototypes such as the one presented by Muckley et al. [10], which performs the sparse sampling by blocking the X-ray source until a projection has to be taken. However, although this approach is not being used yet, there are multiple studies claiming its advantages. In this case, the computational methods needed to perform the reconstructions are algebraic, since the analytical ones do not perform well due to undersampling. They can be either iterative or direct.

When compared to analytical methods to reconstruct the images, new iterative algebraic methods such as SART [11], OS-SART [12], SIRT [13], and LSQR [14,15] greatly reduce the number of projections needed to attain a good image quality. Since they reconstruct the image by solving the system of equations that models the problem through a series of approximations, they do not require that the coefficient matrix has a full-rank to be able to obtain a solution. The disadvantages of the algebraic iterative approach are the following two: The first one is the high computational cost even if they take advantage of the sparsity of the problem. The second one is not being able to determine the final number of iterations needed to obtain a CT study.

In contrast, the use of direct algebraic methods is not so well explored in this area, due to their high computational cost and the fact that they require that the coefficient matrix has full-rank in order to obtain the solution. Because of the latter reason, the number of projections must be higher than when employing the algebraic iterative methods, meaning a smaller dose reduction. In our previous works [16,17], we showed both the feasibility of reconstructing CT images using the direct QR factorization and the extremely high-quality of the reconstructions, which was achieved with a significant dose reduction when compared to the FBP method (up to an 85%).

However, although the coefficient matrix is initially sparse, the problem quickly becomes dense through the factorization process, so the main memory size needed to store the data becomes an important issue for high-resolution images. One of our studies [17] showed that the use of out-of-core (OOC) techniques allows increasing the size (and thus image resolution) of the problem without having a large (and thus expensive) main memory. Since OOC relies on secondary storage, either HDD or SSD, the monetary cost is much smaller, and a problem can be computed as long as there is disk space available to store the data, regardless of its dimensions.

Having determined both the feasibility of using the QR factorization to solve the CT image reconstruction problem and the optimal performance of our OOC method on affordable CPU hardware, the aim of this paper is the adaptation, development, and optimization of the QR OOC method for faster GPU devices. This is an essential step to be able to compete with analytical methods in terms of time performance, since the reconstruction speed in clinical practice must be as high as possible. The best performance reached so far was 1.38 slices per second when performing the computations in CPU [17]. Despite being a good performance for an affordable CPU, several works [18,19] show how iDose, SAFIRE, and FPB can obtain 16, 20, and up to 40 slices per second, respectively.

In this work, we present a GPU implementation of the QR method to solve the CT reconstruction problem, which employs OOC to solve large dimensions problems with high performance on a state-of-the-art hardware platform. The paper contains a solid assessment of high-performance hardware and software, comparing the new GPU implementations to the previous CPU implementation. We show the scalability of our implementation with respect to the number of slices, since computing many more slices increases only slightly the total time. We also show how the OOC approach allows solving systems of dimensions so large that the data would not fit in the GPU memory. Summing up, our new implementation optimized for GPU forms a stable and scalable method for CT reconstruction that is within the time performance and image quality expected in clinical practice and also has the potential to improve the health of the patients since it needs far fewer projections than traditional methods and therefore a lower radiation dose.

The document is organized as follows: Section 2 outlines the theoretical concepts of the method proposed. Section 2.1 introduces the algebraic CT reconstruction process using the QR factorization of the weights matrix, whereas Subsection 2.2 describes in detail the main features of the new software. Section 3 assesses our new method in terms of numerical stability and compares the performance of all the variants developed, as well as the performance of our previous CPU implementation. An analysis of the image quality is also included, comparing several reconstruction methods with our new GPU implementation. Section 4 summarizes and discusses the advantages of the studied method. Section 5 contains the conclusions.

2. Materials and methods

2.1. CT image reconstruction

To reconstruct CT images with an algebraic approach, we model the problem as:

$$AX + N_{\text{elec}} + N_{\text{meas}} = B \tag{1}$$

where $A = \left(a_{i,j}\right) \in \mathbb{R}^{M\times N}$ denotes the so-called system, weights, or coefficient matrix, with dimensions $M \times N$. A models the physical scanner, $a_{i,j}$ being the contribution of the ith ray on the jth pixel. The dimension M is the product of the number of detectors of the CT scanner multiplied by the number of projections or views taken. N denotes the resolution of the image (256×256 pixels, 512×512 pixels, etc.). $B = (B^j)$ is a matrix of $M \times S$ elements, where S is the number of slices to be reconstructed, and B^j denotes the jth column, which corresponds to the jth sinogram. $X = (X^j)$ is a matrix of dimensions $N \times S$, where X^j is the column that will store the reconstructed image corresponding with the jth sinogram. Note that the above formula considers the noise present in the projections, where N_{elec} is the electronic noise and N_{meas} is the noise in the measurements provoked by the scanner. Our work processes simulated projections and focuses on optimizing the reconstruction process of high spatial resolution images based on the QR factorization. Analyzing the effects of the noise in the system is an interesting and complex research that is beyond the goals of this work. Nevertheless, the feasibility of solving the CT image reconstruction problem using the QR factorization with real projections from a micro-CT scanner has already been demonstrated [20]. Although those projections contained noise, the reconstructions were of good quality. Even though no filter was applied, better results than those of other classical methods were obtained, even with a lower spatial resolution. When working with real data, filtering techniques could be applied both before and after the reconstruction method based on the QR factorization. Filtering techniques

should be applied to the sinogram before our reconstruction process as a part of the acquisition process, which modern commercial scanners already perform. Image filters could be applied to the final reconstructions too.

To solve the problem in Eq. (1) without considering the noise, first the QR factorization of A is computed (Eq. (2)), where Q is orthonormal and R is upper triangular. Then, to finally reconstruct the images, Eq. (3) is used. Section 2.2.1 describes in detail these resolution steps. Recall that Q^* is the transpose of Q.

$$A = QR \tag{2}$$

$$X = R^{-1}(Q^*B) \tag{3}$$

A more detailed definition of the scanner parameters used for the simulations can be seen in our previous work [17]. In that paper, we presented the initial approach to the QR method applied to the CT image reconstruction problem, with a CPU implementation that employs out-of-core techniques to solve large-scale systems that do not fit in main memory (RAM). That implementation was optimized to overlap I/O operations with computations so the overall time could be reduced. We encourage the reader to consult the paper to have a better understanding of the techniques we are using.

Although the method was very efficient on medium-cost CPUs, the performance attained was still far from the performance achieved by the more widespread methods. Having already proved the quality and stability of the method, the current aim is to improve its performance by using GPUs and several other optimizations that are explained in the following subsection.

2.2. Algorithms and implementations

Our new algorithm and implementations contain the following contributions:

- The code has been ported to the GPU so that the main computational tasks are performed in the GPU, thus accelerating the full process.
- The block cache management system has been improved to reduce the number of cache misses in order to reduce the number of data transfers.
- Page-locked memory (pinned) is used and assessed in several variants to accelerate data transfers.
- The placement of the storage of the block cache is assessed to determine the best place (main memory, GPU memory, or mirrored in both).

Next, we describe some of the above in more detail.

2.2.1. Porting and optimization of the code for GPU

When considering the full computational process (Eq. (1)), the code comprises six main computational different tasks. In contrast, when considering only the system solving (Eq. (3)), only four main computational tasks are needed. The six computational tasks are the following:

1. Upper triangular system solving.
2. Matrix-matrix product.
3. Computation of a dense QR factorization.
4. Applying a dense QR factorization.
5. Computation of a triangular-dense QR factorization.
6. Applying a triangular-dense QR factorization.

The porting of the first two operations, the upper triangular system solving and the matrix-matrix product, is straightforward by making a call to the cuBLAS `dTRSM` and `dGEMM` subroutines, respectively.

Before describing the porting of the other four tasks, we introduce Householder transformations since they are heavily used in those. A Householder transformation [21] can be defined as:

$$H = I - \tau v v^*, \tag{4}$$

where I is the identity matrix, τ is a scalar, and v is a vector. By choosing appropriate values for τ and v, a Householder transformation can be used to nullify some elements of a column of the coefficient matrix, such as those in the lower triangular part. However, when applying H to the coefficient matrix to nullify some elements, the latter formula employs only matrix-vector operations, which are not so efficient due to the low ratio of the number of floating-point operations to the number of memory accesses in this type of computations. On the other hand, the product of a set of Householder transformations [22,23] $Q = H_1 H_2 \cdots H_b$, $H_i = I - \tau_i v_i v_i^*$ can be combined as:

$$Q = I - YTY^*, \tag{5}$$

where Y is a lower trapezoidal matrix and T is a $b \times b$ upper triangular matrix. The latter formula employs a few matrix-matrix products, which are much more efficient in modern architectures since the ratio of the number of floating-point operations to the number of memory accesses is higher, thus obtaining speeds much closer to the peak speed of the computing device.

Now we are going to describe the porting of the two tasks for computing and for applying the dense QR factorization. Though the cuSOLVER library offers two highly-optimized subroutines for computing and applying the QR factorization, these two subroutines do not return and do not receive, respectively, the T factors from Eq. (5) as arguments. Hence, these factors must be recomputed internally each time the QR factorization is applied. Note that the T factors are computed by using the not so efficient matrix-vector operations. As every QR factorization must be usually applied to many blocks, it is more efficient to save the T factors when computing the dense QR factorizations so that they can be reused when applying the dense QR factorization. Therefore, we have implemented our own version of the dense QR factorization that computes and returns the T factors. Accordingly, we have implemented our version of the applying of a dense QR factorization that receives and uses the T factors, instead of computing them every time. Both implementations employ the cuSOLVER and the cuBLAS libraries.

Finally, we are going to describe the porting of the two tasks for computing and for applying the triangular-dense QR factorization. We have implemented the computation of a triangular-dense QR factorization and its applying since the cuSOLVER library does not contain subroutines for performing these specific operations. Analogously, in this case we also compute and store the T factors during the QR factorization in order to reuse them when updating other blocks. Both implementations also employ the cuSOLVER and the cuBLAS libraries.

2.2.2. Optimization of the block cache management system

In [17] a new out-of-core software that employed a block cache management system was proposed for CPU-based architectures. One of the main features of this system was that it used an LRU (Least-Recently Used) 4-set associative cache. The advantage of this system is that the search to check whether the block is already in the cache is 4 times as fast. The drawback is that blocks can only go to one of the four sets, and when looking for a block to be replaced, only the current set is considered. Another feature was that it only considered square blocks. It could work with rectangular blocks, but when computing maximum sizes and cache sizes only square blocks were considered and the number of entries in the cache was limited by the size of the square blocks.

We propose a new management system that can perform better by changing these two features. We only use an LRU 1-set cache. Searches of blocks are a bit slower, but the LRU method considers all the blocks, thus giving better choices. Besides, the new system is prepared to work with any block size. The number of maximum blocks to be stored is not statically set (except for a very high maximum to avoid a too long list of blocks), and the number of blocks depends only on the memory available in every moment.

Another interesting optimization used in the new system is the storage of the data. As in GPUs every allocation/free operation requires the synchronization of all the GPU threads, a big chunk of memory (main memory or GPU, depending on the variant) is allocated (as large as the cache size to be used) at startup time, and then an own management of allocation/free is used, thus reducing the bottlenecks and accelerating the application.

2.2.3. Page-locked memory

Modern operating systems have memory areas that are page-locked or "pinned", that is, never swapped to secondary storage. The size of this page-locked memory is usually small, but it can be easily defined by the system administrator. As the contents of the page-locked memory cannot be removed from main memory, they usually allow a higher transfer bandwidth and an asynchronous concurrent execution. When transferring between main memory and NVIDIA GPUs, if the main memory buffer is not page-locked, the CUDA driver allocates a pinned block and uses it as an intermediary buffer, thus incurring in a higher overhead.

In some variants, we have used pinned memory to accelerate the transfers between main memory and the GPU, as well as between disk and main memory. Since a block cache size much smaller than main memory usually attains good results, keeping all the block cache storage in pinned memory is nowadays feasible.

2.2.4. Families and variants in the out-of-core implementations

We developed three different families of implementations, according to where the block cache was kept. The `var5` family of implementations kept the block cache in the main memory of the computer. The `var6` family of implementations kept the block cache in the GPU memory. The `var7` family of implementations kept the block cache mirrored in main memory and the GPU memory.

- Family `var5`: This family of implementations keeps the block cache in the main memory of the computer.
  Since this block cache is stored in main memory, the transfer of data between disk and main memory will be overlapped with the transfer of data between main memory and the GPU memory and the computation in the GPU. Therefore, this family will be very beneficial if the transfer time between disk and main memory is comparable to the transfer between disk and the GPU and the computation.
  For this family of implementations we have implemented the following variants:
  - Variant `var5t`: This is the traditional implementation that does not use any block cache. Any input operand is read from disk (into the main memory and then) into the GPU memory every time, and every output operand is written from the GPU memory (into the main memory and then) into the disk every time.
  - Variant `var5c`: To reduce the number of data tranfers, this implementation uses the old block cache management system.
  - Variant `var5d`: To reduce the number of data tranfers, it uses the new block cache management system.
  - Variant `var5v`: Analogous to `var5c` but overlapping of disk I/O and computation is performed to reduce the overall cost.
  - Variant `var5w`: Analogous to `var5d` but overlapping of disk I/O and computation is performed to reduce the overall cost.
  - Variant `var5r`: Analogous to `var5w` but page-locked memory (pinned) is employed to store the block cache.
- Family `var6`: This family of implementations keeps the block cache in the GPU memory.
  Since this block cache is stored in GPU memory, the transfer of data between disk and main memory and between main memory and the GPU memory will be overlapped with the computation in the GPU. Therefore, this family could be very beneficial if the computational time in the GPU is comparable to the transfer between disk and the GPU.
  The `var6t`, `var6c`, `var6d`, `var6v`, `var6w`, `var6p`, and `var6r` variants are analogous to those in the `var5` family. In addition to those, for this family of implementations we have implemented the following variant:
  - Variant `var6z`: As the block cache is stored in the GPU, two CUDA streams are used to overlap computations in the GPU with transfers from disk to main memory to GPU memory (and viceversa transfers). One of the CUDA streams is used for the computations, and the other one is used for the transfers.
- Family `var7`: This family of implementations keeps the block cache mirrored in main memory and the GPU memory.
  Since this block cache is stored in both memories, the transfer of data between disk and main memory will be overlapped with the transfer between main memory and the GPU memory and the computation in the GPU. Therefore, this family could be very beneficial if the computational time in the GPU is comparable to the transfer between disk and the GPU. The advantage of this family with respect to the `var5` family is to reduce some transfers between main memory and the GPU memory.
  For this family of implementations we have implemented analogous variants to those of the `var5` family.

3. Results

In this section, we investigate the speed of our new implementations. In all the experiments double-precision real matrices were processed.

As described before, when solving a linear system of equations $AX = B$, two stages are required: The first one is to compute the QR factorization of A: $A = QR$. The second one is to compute X with the following expression: $X = R^{-1}(Q^*B)$. Note that the first stage can be computed only once since it is independent of B, whereas the second stage must be computed for every image or set of images to be generated. Since the first stage can be computed only once and then reused, unless explicitly stated otherwise, all the following tables and figures report only times of the second stage.

Unless explicitly stated otherwise, all the times shown in the tables and figures of this section are computed as the average of five executions to reduce the effect of variability on the results.

3.1. Experimental setup

Most of the experiments were performed in a computer called `alinna`. It featured two AMD EPYC 7282®processors (base clock at 2.8 GHz), with 32 cores and 512 GiB of RAM in total. It also featured a NVIDIA GPU A100 with 40 GiB of RAM inside this device. Its OS was GNU/Linux (kernel version 4.18.0-240.15.1.el8_3.x86_64). GCC compiler (version 8.3.1 20191121) was used. Intel Math Kernel Library (MKL) 2020.0.1 Product Build 20200208 for Intel(R) 64 architecture was employed. The version of the driver employed in the NVIDIA GPU was 455.32.00 and the version of CUDA was 11.1. In addition to one small Solid-State Drive (SSD) for the operating

Table 1
Residuals $\|AX - B\|_F / \|A\|_F$ computed for both the CPU traditional in-core software (based on Intel MKL code) and our new out-of-core GPU software when solving the system $AX = B$ with a matrix A of dimension $266\,500 \times 262\,144$ and a matrix B with several number of columns.

slices	CPU-based software	new GPU-based out-of-core software
256	$3.04 \cdot 10^{-12}$	$2.04 \cdot 10^{-12}$
512	$1.24 \cdot 10^{-11}$	$7.15 \cdot 10^{-12}$
1024	$1.31 \cdot 10^{-11}$	$7.46 \cdot 10^{-12}$
2048	$1.44 \cdot 10^{-11}$	$7.52 \cdot 10^{-12}$

Table 2
Time in seconds to solve a system $AX = B$ with a random matrix A of dimension $266\,500 \times 262\,144$ and a matrix B with 256 columns for out-of-core variant `var5r` with a block cache of 32 GB for different block sizes.

Block size	Time in QR	Time in solving
5 120	4 692.9	105.7
7 680	3 879.2	135.8
10 240	3 838.6	129.4
12 800	3 127.6	132.2
15 360	3 308.6	129.8
17 920	2 858.3	132.3
20 480	3 015.1	130.5
23 040	2 682.5	137.0
25 600	2 822.1	135.2

system and programming tools, this computer used one Samsung SSD 970 EVO 2TB (Firmware 1B2QEXE7) with an M.2 connector and a capacity of 2 TB to store all the data of the application. According to the Linux operating system `hdparm` tool, the cached read speed was 9642.57 MB/s and the buffered disk read speed was 2499.21 MB/s (the average of ten executions was computed).

Our new implementations were coded with the `libflame` (Release 11104) high-performance library. To perform lower level tasks, our code employed Intel MKL when performing linear algebra computations on the CPU and NVIDIA cuSOLVER and cuBLAS when performing linear algebra computations on the GPU.

Unless explicitly stated otherwise, all the experiments based on the CPUs used all the cores in the computer. When using subroutines of MKL's LAPACK, optimal block sizes determined by that software were employed.

The implementations assessed in this section are described in Section 2.2.4.

3.2. Precision

Table 1 shows the residuals $\|AX - B\|_F / \|A\|_F$ after solving the system $AX = B$ with a matrix A of dimension $266\,500 \times 262\,144$ and several matrices B with different numbers of columns shown in the first column. The second column shows the residuals for the traditional CPU-based in-core software (based on Intel MKL library). The third column shows the residuals for our new out-of-core GPU-based software. In particular, the results of the `var5r` variant are shown. Other variants obtained very similar results. The table shows that our new software obtain a precision slightly better than the CPU-based Intel MKL software.

3.3. Effect of block sizes

Table 2 shows the effect of the block sizes on the time spent in both stages when solving a system $AX = B$ with a random matrix A of dimension $266\,500 \times 262\,144$ and a matrix B with 256 columns. This table reports the times of the `var5r` variant with a 32-GB block cache. Only one execution was run to obtain the times of the computation of the QR factorization shown in this table since the computational cost of the QR factorization is very high and the variability in the obtained times is usually smaller on longer experiments. The first column shows the block size. The second column shows the time to compute the factorization of A: $A = QR$. The third column shows the time to compute the solution: $X = R^{-1}(Q^*B)$. As can be seen, the times for computing the QR factorization (second column) strongly depend on the block size, the best block size being 23 040. In contrast, the times for solving the system (third column) do not depend so much on the block size, the best block size being 5 120. Both the QR factorization and the system solution are important, but the second one is key since the first one can be reused for many system solutions. Therefore, from now on, 10 240 will be used as block size in the rest of experiments since it is a good compromise between both stages and it is the same block size used in previous CPU-based experiments [17].

3.4. Comparison between in-core and OOC

Our new out-of-core codes are able to process matrices so large that they do not fit in main memory and thus must be stored in secondary storage (disks). However, as the matrices are stored in secondary storage, blocks of data must be continuously transferred between this storage and main memory, which can reduce performances if special care is not taken. Therefore, it would be very interesting to compare the performances of both GPU in-core codes and GPU out-of-core codes on the same machine (or similar machines) to assess the effect of the data movement on the performances. Obviously, the development of GPU out-of-core codes is covered in our work, but GPU in-core codes are more problematic. Since current GPU codes require that all the data to be processed is stored in the GPU memory, very large matrices cannot be processed on GPU because the GPU memory size is usually smaller than the main memory. Recently, NVIDIA has provided a new method to process matrices so large that they do not fit in the GPU memory and they are stored in main memory. We assessed this method, but its performances dropped extremely when the size exceeded the GPU memory capacity.

Nevertheless, as a comparison between in-core and out-of-cores would be very interesting to find out if the out-of-core approach actually offers high performances, the only solution was to compare in-core and out-of-core codes based on CPU, instead of GPU. To do that, we employed a similar computer with a main memory large enough to store the data employed in the image processing. Recall that matrix A is of dimension $266\,500 \times 262\,144$ and matrix B is of dimension $266\,500 \times S$, where S is the number of slices. In particular, since the storage of all those data requires about 520 GiB, we employed a computer identical to `alinna` with 768 GiB to assess the in-core codes.

Fig. 1 shows the total and decomposed times for the reconstruction of images of two CPU codes. As said before, though we could not compare the in-core and out-of-core GPU codes, this comparison of CPU codes can be very useful to assess our out-of-core approach. The out-of-core `var4v` is a CPU-based code analogous to the GPU-based `var5v` code. The in-core (all the data fit in main memory) method assessed is a code that we developed to this end based on the efficient and well-known Intel MKL library This code is very simple since it keeps all the data in main memory, but its main restriction is that all data must fit inside main memory; otherwise, it could fail or performances would drop notably. In this case, we included the time (though the plot shows the partial times) to load the data and to save the results. Note that this time could be saved by keeping the data in main memory if many systems are solved in a row. On the other hand, our out-of-core code employs the CPU of the system too. The main advantage is that our code can work with any matrix size, and it only employs 40 GB of main memory (this is a parameter that can be easily reduced or increased). The main disadvantage is the continuous

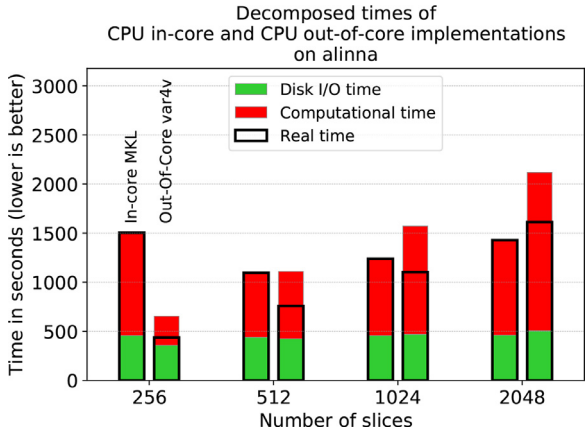

Fig. 1. Decomposed times for CPU in-core and CPU out-of-core codes. For each number of slices, the left bar shows the performance of the in-core MKL code, whereas the right bar shows the performance of the `var4v` out-of-core code with a 40-GB block cache, which is analogous to `var5v`, but executed on the CPU.

Table 3
Number of disk I/O read and write operations and total time in seconds when solving a system $AX = B$ with a matrix A of dimension $266\,500 \times 262\,144$ and a matrix B with two different number of columns (slices) for several out-of-core variants with a cache of 16 GB.

	Cache	256 slices			2048 slices		
Variant	system	Read	Write	Time	Read	Write	Time
`var5t`	no	1858	428	328.4	1858	428	410.0
`var5c`	old	1747	380	332.3	1747	380	508.9
`var5d`	new	1437	189	269.6	1527	246	289.3
`var5v`	old	1747	380	291.5	1747	380	385.0
`var5w`	new	1437	189	215.7	1531	246	198.5

data movement between secondary storage and main memory. As can be noticed, our out-of-core approach is very competitive with the in-core code based on the MKL library. For 256 slices the out-of-core code is much faster because in that case the in-core MKL is very slow. As expected, the real time of our out-of-core method grows linearly (or very close to it) with the number of slices. The real time of the in-core method also grows linearly (or very close to it) with the number of slices except for 256 slices.

3.5. Effect of the block cache management system

Table 3 compares the old and the new block cache management system when solving a system $AX = B$ with a coefficient matrix A of dimension $266\,500 \times 262\,144$ and matrices B with 256 and 2048 columns. The QR factorization is assumed to be already computed, and only the system solving is assessed (Eq. (3)). The out-of-core codes used a block cache of 16 GB. For each number of slices, the table shows the number of disk I/O read operations, the number of disk I/O write operations, and the total time in seconds. Recall that the `var5t` variant does not use any block cache management system, the `var5c` and `var5v` variants employ the old block cache management system, and the `var5d` and `var5w` variants employ the new block cache management system. As you can see, the use of the old block cache management system reduces the number of disk I/O read operations (6% for both number of slices) and disk I/O write operations (11% for both number of slices) of both variants `var5c` and `var5v` with respect to the non-cache variant, whereas only the total time for the `var5v` variant is reduced. On the other hand, the use of the new block cache management system reduces much more the number of disk I/O read operations (23% and 18% for 256 and 2048 slices, respectively) and disk I/O write operations (56% and 43% for 256 and 2048 slices, respectively) of both

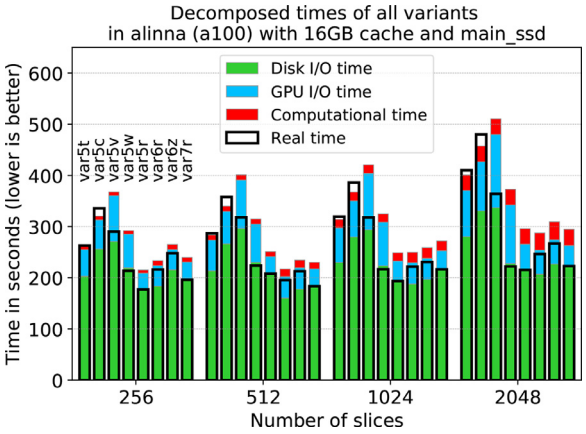

Fig. 2. Decomposed times for the best variants on the main SSD.

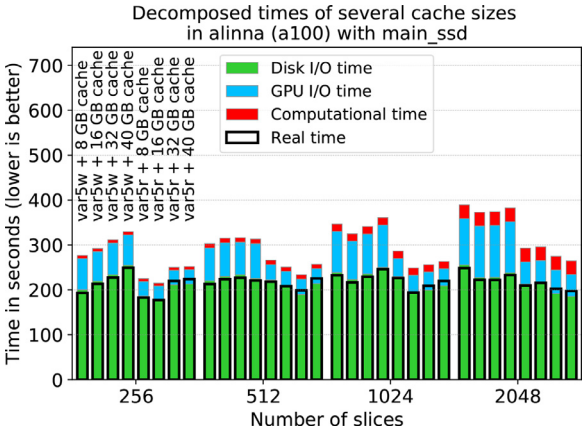

Fig. 3. Decomposed times for several cache sizes on the main SSD.

variants `var5d` and `var5w` with respect to the non-cache variant. The total total time of both `var5d` and `var5w` also dropped. For the latter the time reduction was 34% and 52% for 256 and 2048 slices, respectively.

3.6. Assessment of variants

Fig. 2 shows the total and decomposed times for the reconstruction of images of several variants. Recall that the `var5` variants keep the block cache in main memory, the `var6` variants keep the block cache in the GPU memory, and the `var7` variants keep the block cache mirrored both in main memory and GPU memory. In all the variants assessed for this plot a block cache of 16 GB was used. As can be noticed, the `var5w` variant greatly reduces the disk I/O time of the `var5v` variant since it performs a more efficient use of the block cache. Besides, `var5r` reduces both the disk I/O time and the GPU I/O time of `var5w` even more by using page-locked (pinned) memory. On the other side, the `var7r` variant offer performances slightly lower than `var5r`. The `var6` variants are usually the slowest ones since the block cache is stored in the GPU memory and therefore they overlap disk I/O and GPU I/O with GPU computation. Note the nearly perfect overlapping of some variants such as `var5r` that completely hides the GPU I/O time and the computational time.

3.7. Effect of block cache sizes

To study the effect of the block cache size on performances, Fig. 3 shows the total and decomposed times for the reconstruction of images of two variants with several block cache sizes. As can

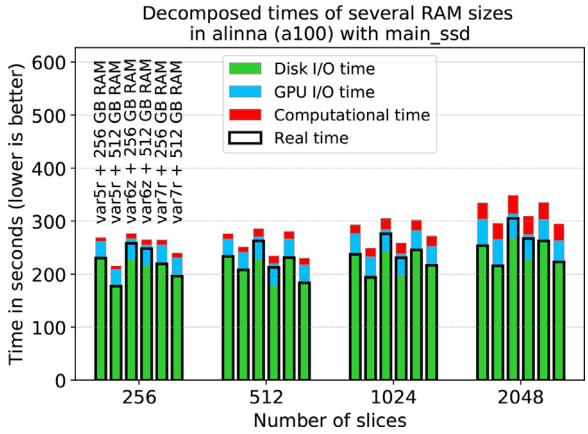

Fig. 4. Decomposed times for several main memory (RAM) sizes on the main SSD.

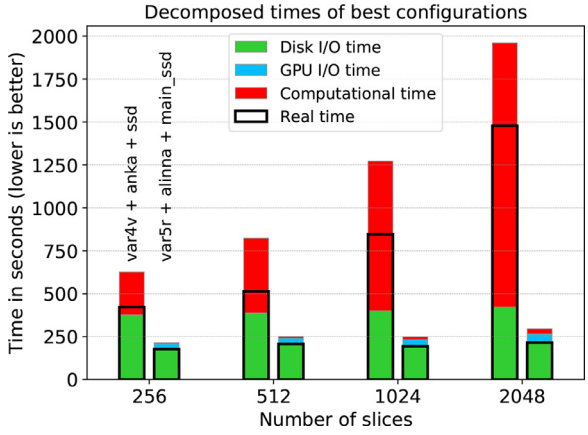

Fig. 5. Decomposed times for the best configurations.

be seen, large block cache sizes affect performances, but the best performances are not always achieved with the largest block cache. The `var5w` variant usually obtain better results with smaller cache sizes. On the other hand, the `var5r` variant obtain the best performances when employing a block cache size of 16 GB for 256 slices, and the largest cache sizes (32 GB and 40 GB) for 2048 slices. This might be produced because the larger the block cache size in main memory, the smaller the space available for disk buffers of the operating system.

3.8. Effect of main memory (RAM) sizes

To study the effect of the size of main memory on performances, Fig. 4 shows the total and decomposed times for the reconstruction of images of two variants with two main memory sizes. All variants in this plot use a block cache of 16 GB. The reader can see that the larger main memory size increases performances for all variants. The cause is that a larger main memory size can be leveraged by the operating system to store more disk buffers, thus reducing the overall I/O cost.

3.9. A comparison of several configurations

Fig. 5 shows the total and decomposed times for the reconstruction of images on several hardware/software configurations. For this plot we have selected the following configurations:

- The first configuration is based on a computer called `anka` with no GPU. Therefore, only the CPU was used in the computations. It featured one Intel i7-7800X®CPU, with 6 physical cores and 128 GiB of RAM in total. Its clock base frequency was 3.50 GHz, and the so-called *Max Turbo Frequency* was 4.00 GHz. In addition to one small SSD for the operating system and programming tools, the computer had one Solid-State Drive (SSD) with an M.2 connector and a capacity of 2 TB to store all the data of the application. This SSD was a Samsung SSD 970 EVO 2TB (Firmware 1B2QEXE7). According to the Linux operating system `hdparm` tool, the read speed of the first one was about 191.43 MB/s, whereas the read speed of the second one was about was 2427.50 MB/s. Its OS was GNU/Linux (kernel version 3.10.0-862.14.4.el7.x86_64). GCC compiler (version 4.8.5 20150623) was used. Intel(R) Math Kernel Library (MKL) Version 2018.0.2 Product Build 20180127 for Intel(R) 64 architecture was employed.
- The second configuration is the out-of-core codes that employ a NVIDIA A100 GPU. See the description above.

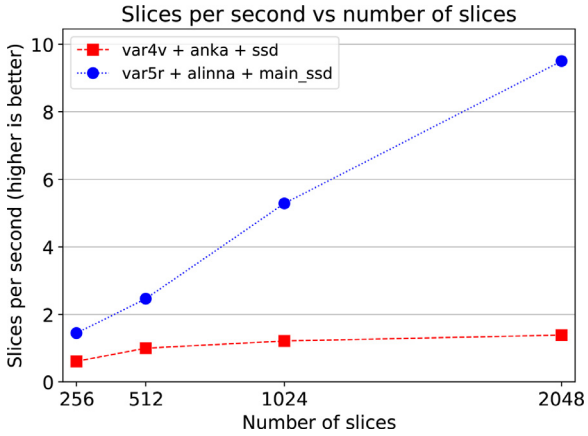

Fig. 6. Slices per second versus several numbers of slices for the best configurations.

Note that both configurations have exactly the same model of SSD (Samsung EVO 970) with the purpose of a fair comparison.

The best available software was employed on both platforms. The `var4v` variant was employed in the `anka` server. It is analogous to `var5v`, but it performs all computations on CPU instead of GPU. The `var5r` variant was employed in the `alinna` server.

The sizes of the block caches for the out-of-core CPU-based software assessed in `anka` and the out-of-core GPU-based software assessed in `alinna` were 32 GB and 16 GB, respectively.

As can be seen, the GPU implementation greatly reduces the computational times of the code based on CPU. For 2048 slices, the GPU implementation is about 7 times faster than the configuration with the CPU i7. Another interesting remark to be made is that the total time of the GPU implementation does not increase much when the number of slices grows from 256 to 2048, thus showing a great scalability with respect to the number of slices. The employment of fast GPUs in our out-of-core code causes the new bottleneck of this application to be the speed of the disks (SSD).

3.10. Speed of image reconstruction

Fig. 6 shows the speed of image reconstruction of same two hardware/software configurations as before. The speed is measured in slices per second. Therefore, unlike the previous plots, higher values are better. As the reader can notice, our new software on the `alinna` server can generate between about 1.5 and 9 slices per second, much better than those attained in the CPU-based server `anka`.

3.11. Image quality

All the images for this study have been selected from the COVID-CT-MD dataset [24], which includes a collection of real CT images obtained from patients with no pathologies, patients with pneumonia due to Covid-19, and patients with community-acquired pneumonia (CAP). The images have been re-projected with Joseph's [25] method in order to simulate the sinograms and then reconstruct them using the QR method. The CT scanner simulated has 1025 detectors, and the number of projections needed to have a full-rank system matrix is 260. The chosen image resolution of the reconstructions is 512×512 pixels. More details on the geometry of the scanner can be found in our previous work [17].

Three image quality metrics have been used to test the quality of the reconstructed images. The first one is the Mean Absolute Error (MAE). It denotes the average difference of the pixels in absolute value all over the image. See Eq. (6), where M and N are the numbers of rows and columns in pixels, I_0 is the reference image, and I is the reconstructed image.

The second one is called Peak Signal-to-Noise Ratio (PSNR), and it measures the level of image noise (see Eq. (7)). In the equation, MAX represents the maximum value that a pixel can take.

Finally, the third one is the Structural Similarity Index (SSIM). It is a perceptual metric that measures the level of conservation of the internal structures and edges of the images (see Eq. (8)). It is applied through pairs of windows of fixed size, and the difference between two windows x and y corresponding to the two images to be compared is calculated. In this equation, μ_x and μ_y are the average values of the respective window x and y, σ_x^2 and σ_y^2 are the variances, σ_{xy} is the covariance between the two windows, and c_1 and c_2 are two stabilizing variables dependent on the dynamic range of the image. More information about these quality metrics can be found in [26]. Specific parameters for each metric are more detailed in [17].

$$\text{MAE} = \frac{1}{MN}\sum_{i=0}^{M-1}\sum_{j=0}^{N-1}|I_0(i,j) - I(i,j)| \tag{6}$$

$$\text{PSNR} = 10\,\log_{10}\frac{\text{MAX}(I_0)^2}{\text{MSE}}, \quad \text{with}$$

$$\text{MSE} = \frac{1}{MN}\sum_{i=0}^{M-1}\sum_{j=0}^{N-1}(I_0(i,j) - I(i,j))^2 \tag{7}$$

$$\text{SSIM} = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{x,y} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)} \tag{8}$$

In order to assess the quality of the proposed GPU-based implementation, the same reconstructions have been performed using two other techniques. The first one is the Least Squares QR (LSQR) method, which has been combined with both the Soft-Thresholding-Filter (STF) and the FISTA acceleration [27,28]. It is an iterative algebraic method that can thus deal with rank-deficient problems and that was employed for solving the same system of equations as that solved by the QR method (Eq. (1)) using 260 projections. As was shown [27], the LSQR attained both better quality and significantly lower computational times than the SART method, which is one of the most widely methods used in algebraic reconstructions. For this reason, it was selected for this comparison instead of SART.

The second method selected is the classical analytical method Filtered Back-Projection (FBP) using the Ram-Lak filter [9], which continues to be the most common reconstruction technique. Although nowadays there are state-of-the-art iterative reconstruction methods (IR) that take into account the scanner information to improve the FBP reconstructions, they are complex methods provided by manufacturers via embedded software in their systems.

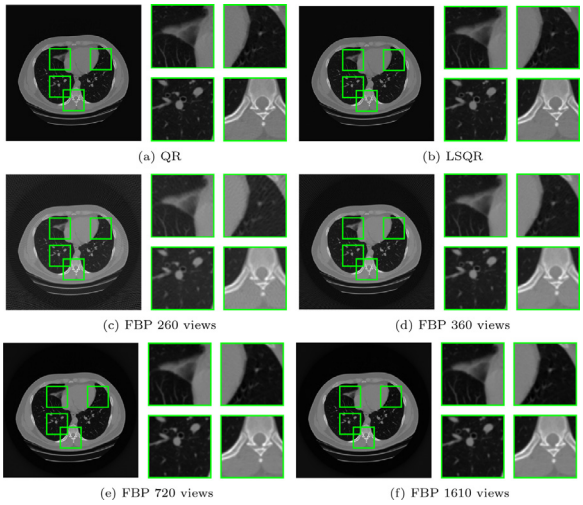

Fig. 7. No pathology.

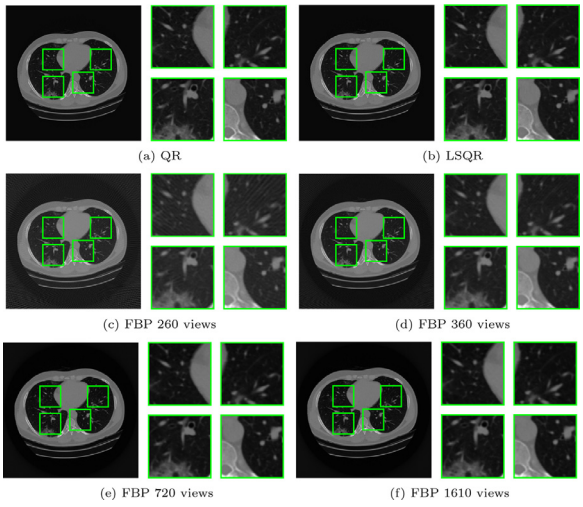

Fig. 8. Covid pneumonia.

Therefore, as we do not have access of any of this software, we have not been able to include a comparison with IR methods here. The FBP method does not perform optimally when the number of projections is small due to undersampling. The minimum number of views is determined by the Nyquist-Shannon sampling theorem [29], and for this particular problem we need to employ around 1600 projections to obtain a good quality, as was previously shown [17]. Hence, when compared to the analytical method FBP, the QR method could mean an 85% reduction of the number of projections needed since it only requires 260 out of 1600. This is a significant reduction, and even if iterative algebraic methods need fewer projections, they also imply a higher error, as we will show in the results.

Figs. 7, 8, and 9 show the images of three reconstructions, corresponding to each of the type of case included in the dataset: healthy patients, patients with Covid-19, and patients with CAP.

Table 4 shows the results of the image quality metrics applied to these reconstructions. The FBP method employed 260, 360, 720, and 1610 projections. It is worth mentioning that the metrics have been calculated using the images with Hounsfield Units (HU) values.

As can be seen in this table, the reconstructions using the QR method attain the highest quality, and they can be considered identical to the reference images in every case, with a perfect SSIM

Table 4
Results of the image quality metrics applied to the reconstructions corresponding to three selected CT images: no pathology, Covid pneumonia, and community-acquired pneumonia (CAP).

		No pathology			Covid			CAP		
Method	# Projec.	MAE	PSNR	SSIM	MAE	PSNR	SSIM	MAE	PSNR	SSIM
QR	260	$3.5 \cdot 10^{-8}$	207.22	1	$3.6 \cdot 10^{-8}$	207.30	1	$3.6 \cdot 10^{-8}$	206.54	1
LSQR	260	2.65	50.05	0.97	3.09	40.01	0.97	9.74	36.22	0.92
FBP	260	45.16	26.37	0.33	46.76	26.48	0.34	42.85	25.41	0.38
FBP	360	34.81	28.28	0.43	36.08	28.50	0.43	31.75	27.77	0.48
FBP	720	27.63	29.76	0.75	27.98	30.19	0.74	24.28	29.37	0.80
FBP	1610	27.06	29.79	0.86	27.41	30.24	0.86	23.82	29.41	0.90

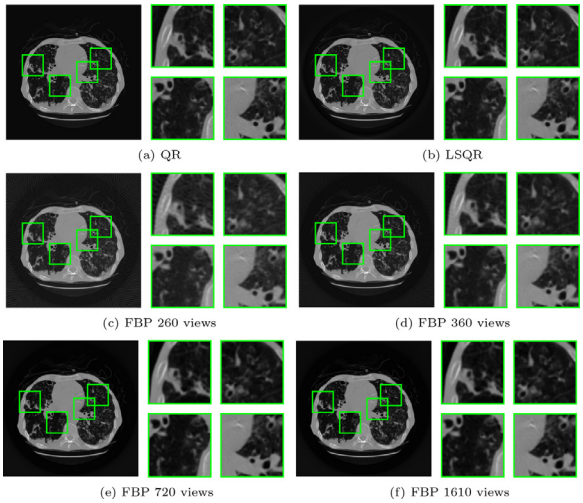

Fig. 9. Community-acquired pneumonia.

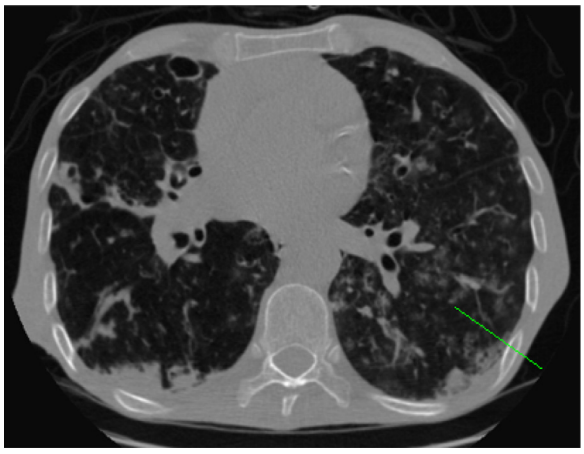

Fig. 10. Selected segment of the CAP image.

and a very low MAE. For this reason, the reference images for this case are not shown.

The second best quality results are obtained by LSQR. Although the SSIM does not reach 1, the error is hardly perceived by the human eye, as can be seen in Figs. 7b, 8 b, and 9 b. However, the metrics show that the quality is not as high as the obtained with QR, and it is mainly due to the oversmoothing effect of this method, as can be seen in Fig. 9b, in which the image loses some texture compared to the reference image. Attaining a MAE ranging from 2.65 to 9.74 HU could be detrimental for the application of postprocessing techniques to the images, such as image segmentation or automatic detection of anomalies through artificial intelligence.

The reconstructions by the FBP methods are the ones with the poorest quality. If FBP is used with the same number of projections as that required by the algebraic methods (260), the error and noise obtained is very high, losing a third part of the internal structures information. Fig. 8c clearly shows the streak artifacts, and how they affect the structures of the image. When the number of projections is increased, the quality improves accordingly, but in every reconstruction slightly blurry edges can be noticed (reflected by the SSIM results), as well as a slight change in the grey scale of the images, compared to the reference. Even when using the optimal number of views the quality does not match that obtained by the algebraic methods. The best reconstruction has a MAE of 23.82 HU, much worse than the 9.74 HU attained by LSQR.

In order to show how the intensity of the pixels fluctuate with all the methods, Fig. 11 shows the intensity profile along a chosen segment of the image corresponding to a community-acquired pneumonia case (the segment is shown in Fig. 10, displayed in green). It can be seen how the profiles of both the QR and the LSQR are very similar, the QR profile being identical to the reference image profile, whereas FBP with 260 views is very different from the original values. The reconstruction by FPB with 1610 projections does not fluctuate that much, but it still shows error. The FBP with 360 and 720 projections are not displayed, but the error ranges between the best and the worst FBP cases displayed.

4. Discussion

In this paper, we present an optimized implementation of the QR method applied to the CT image reconstruction problem. Our software combines the use of GPU computing with out-of-core techniques to attain high performance when solving large problems that do not fit in the main memory of the computer.

In this new implementation, we have improved the block cache management system and we have assessed several configurations regarding the placement of the cache. The three options evaluated are the following: keeping the blocks in main memory, in GPU memory, or mirrored in both memories. The best performance is attained when employing the main memory of the computer to store the block cache making use of pinned memory to reduce the transfer time among disk, main memory, and GPU memory.

The results obtained show that our new GPU version clearly outperforms the previous implementation on the CPU, with speedups of the reconstruction step ranging from 2.5 to 6.5 for 256 and 2048 slices, respectively. The speed in terms of slices per second has increased from 0.6 slices per second on CPU to 1.7 slices per second on GPU for 256 slices, and it has increased from 1.4 slices per second on CPU to 9 slices per second on GPU for 2048 slices. The improvement factor is significant, achieving a performance that can be competitive with the IR methods used by the manufacturers.

For this reason, once the sparse sampling scheme is introduced in clinical practice, we believe our approach could be a robust and efficient method for performing the reconstructions with a reduced

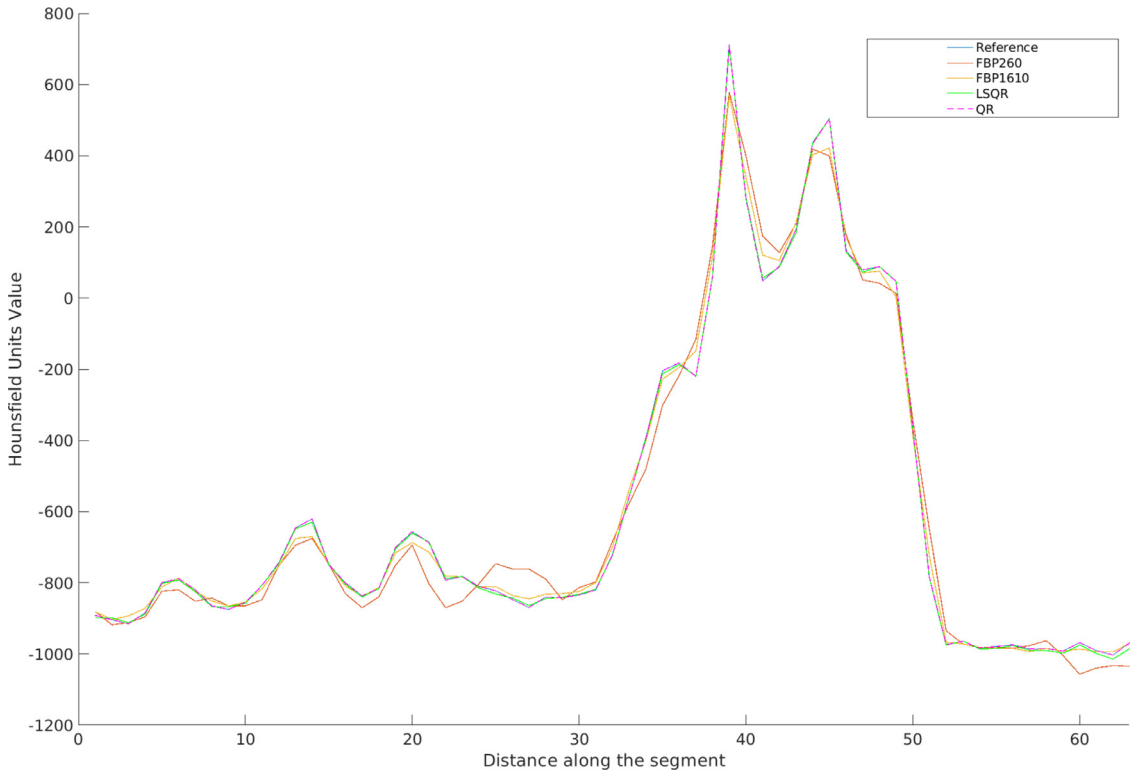

Fig. 11. Intensity profile along a segment of the CAP image.

number of projections, which implies lower radiation doses for patients. The image quality obtained with the QR method is extremely high when the matrix has full rank, since it obtains an exact solution, which makes it more reliable than algebraic iterative methods. As shown in the quality analysis, the performance of the LSQR method is good, but the images still contain noise and they suffer from oversmoothing, which can cause to lose relevant information. The more complex the internal structures are, the lower the quality is. This can be easly appreciated in the reconstructions: The simpler one (no pathology) achieves good quality with LSQR, the one corresponding to bilateral community-acquired pneumonia has lower quality since it is more complex, and the quality corresponding to Covid-19 is between the previous two. However, employing an iterative algebraic method such as LSQR could also noticeably reduce the number of projections needed, since it can process rank-deficient matrices. A study [28] showed that the LSQR method can obtain good reconstructions even with a lower number of projections. In contrast, another study [30] showed that the quality of the QR decreases when the matrix does not have full rank. For a 512×512 resolution, it attains good quality using 30 noise-free projections of a mathematical phantom. Unlike the QR method, it is worth mentioning that the LSQR required an in-depth search of the optimal parameters to obtain this quality. Therefore, the two algebraic reconstruction alternatives could co-exist, with the direct approach oriented to get the best image quality, ensuring a minimum error and a more robust process, and the iterative one oriented to minimize the radiation dose even if both the error and the variability (depending on the parameters) is higher. The decision on which one to employ would depend on the patients and their particular needs.

In addition, in our experimental study, we show that in the GPU implementation the number of slices to be reconstructed does not strongly affect the overall reconstruction time. In contrast, in the CPU implementation the increase of the number of slices meant a strong increase in the total time. Therefore, this optimization is very suitable for full-body CTs, as well as for reconstructing multiple studies at once (batch processing). For instance, the total time needed to reconstruct a study with 256 slices is 2.5 min, whereas the time needed to reconstruct 2048 slices (8 times as large) is only 3.8 min.

Besides, since now the bottleneck is the SSD performance, the overall time could be further reduced with new PCIe 4.0 Solid-State Drives, which could double the reading speed and improve the writing speed by a 1.5 factor approximately. Another interesting option would be a high-performance RAID system with current SSD disks that increased the read and write speeds. Though the speed of our GPU method is close to that of commercial methods, both types of new hardware would allow faster reconstructions, making our method even more competitive.

Finally, it is worth mentioning that the numerical stability of this method allows us to increase the image resolution and thus the size of the problem with the same hardware as long as there is enough storage space without suffering from image quality loss, which we intend to do in the future. In addition, although we are now modelling the problem as a 2D multi-slice reconstruction process, it would be possible to adapt the method to 3D approaches such as Cone-Beam CT (CBCT). Some works [31–33] employ algebraic methods to reconstruct CBCT volumes by modelling the system matrix using voxels instead of pixels. Other works [20] solve the CBCT system by using the QR method, but for small micro-CT images. Switching to CBCT would mean a larger matrix and thus a larger equations system, but by using OOC techniques it would not pose a problem in terms of memory requirements.

5. Conclusions

Our new implementation of the QR method provides an efficient, scalable, and robust approach to solving the CT image reconstruction problem with a direct algebraic method on GPUs when the number of projections is reduced to lower the exposure time to

the X-rays. The method has been thoroughly optimized to employ an optimal block cache system in order to reduce the data transfer times, and with the high performance delivered by the GPU for this out-of-core approach, the overall reconstruction time has been significantly reduced.

Although the monetary cost of the hardware we employ is about one order of magnitude higher than with our CPU version, since top-of-the-line GPU cards are more expensive, we consider the equipment is still affordable for this kind of medical application.

Future investigations should consider using real projections, such as those from the open-access library of CT patient projection data provided by the Mayo Clinic [34]. The method presented in this paper does not contemplate noise in the sinograms, so the quality of the reconstructions may be lower when working with real data. Thus, it would be very interesting to assess the effect of noise in the resulting images and to integrate filtering techniques on our reconstruction process.

Declaration of Competing Interest

The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be constructed as a potential conflict of interest.

Acknowledgements

This research has been supported by the "Universitat Politècnica de València", "Generalitat Valenciana" under PROMETEO/2018/035 and ACIF/2017/075, co-financed by FEDER and FSE funds, and the "Spanish Ministry of Science, Innovation and Universities" under Grant RTI2018-098156-B-C54 co-financed by FEDER funds. The authors would also like to thank Francisco D. Igual (Universidad Complutense de Madrid) for granting access to the `volta1` server.

References

[1] J.M. Meulepas, C.M. Ronckers, A.M. Smets, R.A. Nievelstein, P. Gradowska, C. Lee, A. Jahnen, M. van Straten, M.-C.Y. de Wit, B. Zonnenberg, et al., Radiation exposure from pediatric CT scans and subsequent cancer risk in the Netherlands, J. Natl. Cancer Inst. 111 (3) (2019) 256–263.

[2] J.-Y. Hong, K. Han, J.-H. Jung, J.S. Kim, Association of exposure to diagnostic low-dose ionizing radiation with risk of cancer among youths in South Korea, JAMA Netw. Open 2 (9) (2019). e1910584–e1910584

[3] L. Krille, S. Dreger, R. Schindel, T. Albrecht, M. Asmussen, J. Barkhausen, J. Berthold, A. Chavan, C. Claussen, M. Forsting, et al., Risk of cancer incidence before the age of 15 years after exposure to ionising radiation from computed tomography: results from a German cohort study, Radiat. Environ. Biophys. 54 (1) (2015) 1–12.

[4] M.S. Pearce, J.A. Salotti, M.P. Little, K. McHugh, C. Lee, K.P. Kim, N.L. Howe, C.M. Ronckers, P. Rajaraman, A.W. Craft, et al., Radiation exposure from ct scans in childhood and subsequent risk of leukaemia and brain tumours: a retrospective cohort study, Lancet 380 (9840) (2012) 499–505.

[5] C.H. Schultz, R. Fairley, L.S.-L. Murphy, M. Doss, The risk of cancer from CT scans and other sources of low-dose radiation: a critical appraisal of methodologic quality, Prehosp. Disaster Med. 35 (1) (2020) 3–16, doi:10.1017/S1049023X1900520X.

[6] D. Lee, S. Choi, H. Lee, D. Kim, H.-J. Kim, Quantitative evaluation of anatomical noise in chest digital tomosynthesis, digital radiography, and computed tomography, J. Instrum. 12 (04) (2017) T04006.

[7] T. Kubo, Y. Ohno, M. Nishino, P.-J. Lin, S. Gautam, H.-U. Kauczor, H. Hatabu, iLEAD Study Group, et al., Low dose chest ct protocol (50 mAs) as a routine protocol for comprehensive assessment of intrathoracic abnormality, Eur. J. Radiol. Open 3 (2016) 86–94.

[8] A.N. Khan, F. Khosa, W. Shuaib, K. Nasir, R. Blankstein, M. Clouse, Effect of tube voltage (100 vs. 120 kVp) on radiation dose and image quality using prospective gating 320 row multi-detector computed tomography angiography, J. Clin. Imaging Sci. 3 (2013).

[9] X. Tang, J. Hsieh, R.A. Nilsen, S. Dutta, D. Samsonov, A. Hagiwara, A three-dimensional-weighted cone beam filtered backprojection (CB-FBP) algorithm for image reconstruction in volumetric CT helical scanning, Phys. Med. Biol. 51 (4) (2006) 855.

[10] M.J. Muckley, B. Chen, T. Vahle, T. O'Donnell, F. Knoll, A.D. Sodickson, D.K. Sodickson, R. Otazo, Image reconstruction for interrupted-beam X-ray CT on diagnostic clinical scanners, Phys. Med. Biol. 64 (15) (2019) 155007.

[11] M. Jiang, G. Wang, Convergence of the simultaneous algebraic reconstruction technique (SART), IEEE Trans. Image Process. 12 (8) (2003) 957–961.

[12] G. Wang, M. Jiang, Ordered-subset simultaneous algebraic reconstruction techniques (OS-SART), J. X-ray Sci. Technol. 12 (3) (2004) 169–177.

[13] J. Gregor, T. Benson, Computational analysis and improvement of SIRT, IEEE Trans. Med. Imaging 27 (7) (2008) 918–924.

[14] E. Parcero, L. Flores, M.G. Sánchez, V. Vidal, G. Verdú, Impact of view reduction in ct on radiation dose for patients, Radiat. Phys. Chem. 137 (2017) 173–175.

[15] L.A. Flores, V. Vidal, P. Mayo, F. Rodenas, G. Verdú, Parallel CT image reconstruction based on GPUs, Radiat. Phys. Chem. 95 (2014) 247–250.

[16] M. Chillarón, V. Vidal, G. Verdú, CT image reconstruction with SuiteSparseQR factorization package, Radiat. Phys. Chem. (2019), doi:10.1016/j.radphyschem.2019.04.039.

[17] M. Chillarón, G. Quintana-Ortí, V. Vidal, G. Verdú, Computed tomography medical image reconstruction on affordable equipment by using out-of-core techniques, Comput. Methods Programs Biomed. 193 (2020) 105488, doi:10.1016/j.cmpb.2020.105488.

[18] A. Moscariello, R.A. Takx, U.J. Schoepf, M. Renker, P.L. Zwerner, T.X. O'Brien, T. Allmendinger, S. Vogt, B. Schmidt, G. Savino, et al., Coronary CT angiography: image quality, diagnostic accuracy, and potential for radiation dose reduction using a novel iterative image reconstruction technique-comparison with traditional filtered back projection, Eur. Radiol. 21 (10) (2011) 2130–2138.

[19] Y. Funama, K. Taguchi, D. Utsunomiya, S. Oda, Y. Yanaga, Y. Yamashita, K. Awai, Combination of a low tube voltage technique with the hybrid iterative reconstruction (iDose) algorithm at coronary CT angiography, J. Comput. Assist. Tomogr. 35 (4) (2011) 480.

[20] M.J. Rodríguez-Alvarez, F. Sánchez, A. Soriano, L. Moliner, S. Sánchez, J.M. Benlloch, QR-factorization algorithm for computed tomography (CT): comparison with FDK and conjugate gradient (CG) algorithms, IEEE Trans. Radiat. Plasma Med.Sci. 2 (5) (2018) 459–469.

[21] G.H. Golub, C.F. van Loan, Matrix Computations, 4th ed., JHU Press, 2013.

[22] C. Bischof, C. Van Loan, The wy representation for products of householder matrices, SIAM J. Sci. Stat.Comput. 8 (1) (1987) s2–s13.

[23] T. Joffrain, T.M. Low, E.S. Quintana-Ortí, R. van de Geijn, F.G. Van Zee, Accumulating householder transformations, revisited, ACM Trans. Math. Softw. 32 (2) (2006) 169–179, doi:10.1145/1141885.1141886.

[24] P. Afshar, S. Heidarian, N. Enshaei, F. Naderkhani, M.J. Rafiee, A. Oikonomou, F.B. Fard, K. Samimi, K.N. Plataniotis, A. Mohammadi, COVID-CT-MD, COVID-19 computed tomography scan dataset applicable in machine learning and deep learning, Sci. Data 8 (1) (2021) 1–8.

[25] P. Joseph, An improved algorithm for reprojecting rays through pixel images, IEEE Trans. Med. Imaging 1 (3) (1982) 192–196.

[26] A. Hore, D. Ziou, Image quality metrics: PSNR vs. SSIM, in: 2010 20th International Conference on Pattern Recognition, IEEE, 2010, pp. 2366–2369, doi:10.1109/ICPR.2010.579.

[27] L. Flores, V. Vidal, G. Verdú, Iterative reconstruction from few-view projections, Procedia Comput. Sci. 51 (2015) 703–712.

[28] M. Chillarón, V. Vidal, D. Segrelles, I. Blanquer, G. Verdú, Combining grid computing and docker containers for the study and parametrization of CT image reconstruction methods, Procedia Comput. Sci. 108 (2017) 1195–1204, doi:10.1016/j.procs.2017.05.065. International Conference on Computational Science, ICCS 2017, 12–14 June 2017, Zurich, Switzerland

[29] A.C. Kak, M. Slaney, Principles of Computerized Tomographic Imaging, Society for Industrial and Applied Mathematics, 2001, doi:10.1137/1.9780898719277.

[30] M. Chillarón, V. Vidal, G. Verdú, J. Arnal, CT medical imaging reconstruction using direct algebraic methods with few projections, in: Computational Science – ICCS 2018, Springer International Publishing, Cham, 2018, pp. 334–346.

[31] M.A. Al-masni, M.A. Al-antari, M.K. Metwally, Y.M. Kadah, S.-M. Han, T.-S. Kim, A rapid algebraic 3D volume image reconstruction technique for cone beam computed tomography, Biocybern. Biomed. Eng. 37 (4) (2017) 619–629.

[32] H.C. Lee, B. Song, J.S. Kim, J.J. Jung, H.H. Li, S. Mutic, J.C. Park, Variable step size methods for solving simultaneous algebraic reconstruction technique (SART)-type CBCT reconstructions, Oncotarget 8 (20) (2017) 33827.

[33] W. Qiu, T. Pengpan, N. Smith, M. Soleimani, Evaluating iterative algebraic algorithms in terms of convergence and image quality for cone beam CT, Comput. Methods Programs Biomed. 109 (3) (2013) 313–322.

[34] T.R. Moen, B. Chen, D.R. Holmes III, X. Duan, Z. Yu, L. Yu, S. Leng, J.G. Fletcher, C.H. McCollough, Low-dose CT image and projection dataset, Med. Phys. 48 (2) (2021) 902–911.