GPU-accelerated Computed Laminography with Application to Non-destructive Testing

Michael MAISL 1, Lukas MARSALEK 2, Christian SCHORR 1, Jan HORACEK 2, Philipp SLUSALLEK 3,4

1 Fraunhofer EZRT, Saarbruecken, Germany, e-mail: christian.schorr@izfp.fraunhofer.de
2 Eyen SE, Prague, Czech Republic, e-mail: lukas.marsalek@eyen.se
3 Saarland University, Saarbruecken, Germany
4 DFKI GmbH, Saarbruecken, Germany, e-mail: slusallek@dfki.de

Abstract

Computed tomography (CT) is a very powerful tool in medicine and non-destructive testing but it is unsuitable for planar objects like printed circuit boards or fiber reinforced plastics sheets, due to their strongly varying penetration lengths and spatial restrictions during the scan. A solution can be found in the use of computed laminography (CL), a technique where the object is irradiated by an oblique angle, thereby circumventing the problems arising in CT.

The innovative scanner system CLARA (Computed laminography and radioscopy device) realizes this geometry in a new and efficient way, compared to existing industrial systems. Instead of four translational axes, CLARA only needs one rotational axis, greatly reducing both the costs and the calibration errors.

Due to the limited amount of angular coverage and the specific geometric setup, filtered back projection methods used in CT cannot be employed for the reconstruction of laminographic projections. More flexible iterative algorithms like SART (simultaneous algebraic reconstruction technique) provide an answer to this challenge and also allow to incorporate a priori knowledge about the object to increase the reconstruction quality. The drawback of these algorithms lies in their high computational demands, resulting in typical reconstruction times in the order of hours. For certain practical applications this may be too time-consuming and therefore an acceleration of the algorithms is necessary.

Compared to desktop central processing units (CPUs), modern graphics processing units (GPUs) contain an order of magnitude more cores designed specifically for massive data parallel processing. By designing new algorithms to fully exploit the GPU architecture, reconstruction times can be reduced from hours on a CPU to mere minutes on the GPU, thus allowing the practical use of computed laminography in industrial settings. Areas of application include the inspection of printed circuit boards, fiber reinforced plastics parts of wind energy plants and cars, as well as the general testing of sheet-like objects.

We explain the advantages of the CLARA scanner and demonstrate the feasibility of computed laminography by means of CLARA results of industrially relevant inspection challenges using GPU-enhanced algorithms.

Keywords: Computed laminography, CLARA, GPU acceleration, iterative reconstruction methods

1. Computed Laminography

Computed tomography (CT) is a well-established and widely used non-destructive inspection method for the analysis of the interior structure of objects. However, often one is faced with the challenge of inspecting planar objects like sheets of fiber reinforced plastics or printed circuit boards. Applying standard reconstruction methods for circular or helical sampling to planar objects, two fundamental problems arise: impenetrability in longitudinal direction and collision risks between X-ray source and object at high magnifications.

During a CT, the object is rotated by 360° while being irradiated. Planar objects are challenging since they exhibit very different irradiation lengths. In normal direction to the
surface, absorption is very much lower than in the longitudinal direction. Trying to compensate for this by increasing the energy of the X-rays, one automatically reduces contrast and geometrical resolution, thereby possibly rendering the reconstruction useless.

The opening angle of the X-ray source allows for a variation of magnification by changing the distance between X-ray source and object. In this way, small object features can be inspected in detail. Especially planar objects with very fine structures can require such a high magnification, that the required source-detector distance gets too small to permit a full $360^\circ$ rotation without risking a collision between the source and the object. Circumventing this problem by increasing the source-object and source-detector distances, while maintaining the desired magnification ratio results in a severely limited opening angle. This in turn restricts the field of view, making multiple scans necessary to cover the entire area of interest.

Computed Laminography (CL) can solve these problems. In standard CT geometries the X-ray source and detector are perpendicular to each other and the axis of rotation and a full $360^\circ$ coverage is necessary. In contrast, CL can also work with a limited angular range of less than $90^\circ$ (Swing Laminography) or completely dispense with the traditional setup and use linear translational (translation CL), planar translational (classic CL) or rotational geometries (CLARA (Computed Laminography And RAdiography) [1, 2, 3]. The advantage of all these trajectories lies in their ability to place the object close enough to the source to achieve the desired magnification without colliding with the X-ray tube. Additionally, most of these geometries permit a constant oblique irradiation angle throughout the entire measurement. This eliminates the problem of widely differing object thicknesses with all the drawbacks mentioned above. In this paper we focus on the CLARA geometry, which has been realized in an experimental CL-scanner at Fraunhofer IZFP in Saarbrücken, Germany [4].

![Fig. 1. Schema of the CLARA geometry](image)

The object is situated on a swing bearing, allowing the source beneath to irradiate it under an oblique laminography angle $\lambda$. The detector is positioned perpendicular to the central axis of the source. For the measurement, the object is rotated $360^\circ$ around the rotation axis, while resting upon the swing bearing. This setup allows to achieve a very high magnification and spatial resolution, while avoiding the usual CT drawbacks.

Due to the special CL geometry, standard CT reconstruction algorithms, like the Feldkamp method are no longer applicable for mathematical reasons. Therefore, a more flexible iterative algorithm like SART (Simultaneous algebraic reconstruction technique) [7] is used. It computes the reconstruction of the density distribution of an object by iteratively solving a system of linear equations. The physical process of a CL measurement is modeled as a matrix-vector equation and solved iteratively:
Fig. 2. Model of the physical measurement process as represented by the SART algorithm. Radiation is modelled as a ray from the source to each detector pixel along passing through a volume of cubic voxels representing the object’s density.

Be \( v \) a volume consisting of \( N \) cubic voxels \( j \) with constant values \( v_j \) which represent the absorption coefficients of the object. Furthermore be \( p \) the vector of measured rays of dimension \( M \) with \( p_i \) the ray sum along the ray \( i \) passing through \( v \). Each \( p_i \) thus represents one measured detector pixel. Then the relation between \( v \) and \( p \) can be expressed by the following equation:

\[
Wv = p, \quad \sum_{j=1}^{N} w_{ij} v_j = p_i, i = 1, 2, ..., M
\]  

(1)

where \( W \) is a matrix of dimension \( M \times N \) with entries \( w_{ij} \) representing the weighting coefficient in voxel \( j \) along ray \( i \). \( w_{ij} \) can be interpreted as the length of ray \( p_i \) through voxel \( j \).

Be \( v_j^{(k)} \) the value of voxel \( j \) after \( k \) projections, \( \lambda \) a relaxation factor and \( P_\varphi \) the projection belonging to angle \( \varphi \). Then the SART algorithm is defined as:

\[
v_j^{(k+1)} = v_j^{(k)} + \lambda \left( \sum_{l=1}^{M} \frac{w_{lj}}{w_{lj}^{\neq 0}} \left( p_l - \sum_{n=1}^{N} \frac{w_{ln} v_n^{(k)}}{w_{ln}^{\neq 0}} \right) \right) w_{ij}
\]

(2)

This formulation leads to the following algorithm. For each projection \( P_\varphi \) in direction \( \varphi \) do:

a. For all rays \( p_i \in P_\varphi \) do:
   i. “Simulate” the process of CL by computing an estimate \( \tilde{p}_i \) of \( p_i \) from the current volume estimate \( v^{(k)} \) by summing and weighting the voxel
values \( v_j^{(k)} \) along the corresponding ray \( i \), i.e. evaluate one row of the matrix in Equation 1. This step is referred to as the forward projection.

\[ b. \text{Compute the error } e_i \text{ of the estimate } \bar{P}_i \text{ by subtracting it from } p_i \]

\[ \text{ii. For all voxels } v_j^{(k)} , j=1...N \text{ in the volume do:} \]

\[ \text{i. Sum and weight the errors } e_i \text{ from all the rays in the projection } P_y \text{ back into voxel } v_j^{(k)}, \text{ forming a new estimate } v_j^{(k+1)} \text{ of the voxel’s value. This part of the algorithm is referred to as the back projection.} \]

An additional advantage of the algebraic reconstruction techniques in general is the possible integration of geometrical a priori information to increase the reconstruction quality \([5, 6]\). If knowledge about the objects contours is available in form of a CAD model or from another inspection method like ultrasound or thermography, then a fusion of CL data and a priori information can be incorporated into SART, leading to a vastly improved defect detectability.

2. GPU Architecture and parallelization

To be able to run computation-heavy algorithms such as iterative reconstruction methods fast enough for common industrial scenarios, the speed of current CPUs would have to be increased significantly. However, several fundamental technological issues do not allow to scale the performance of a single core much beyond what current hardware can already achieve. Multicore architectures are one of the solutions to this scaling problem.

The approach of multicore architectures is to stack a large number of simple processing units spatially close to each and let them execute parts of the program, the work items, in parallel. Part of this parallel execution is handled by the hardware automatically and part of it is delegated to the programmer, necessitating a suitable parallel algorithm (re)design.

![Fig. 3. An example block diagram of contemporary GPU (NVIDIA Kepler Architecture, adapted from [9]).](image)

The heart of the GPU chip (A) is composed of complex processing units, so called multiprocessors (green, SMX). These are accompanied by a fast memory (blue) and scheduling (orange) logic. Each multiprocessor (B) then contains hundreds of individual computing cores. Altogether the GPU features thousands of processing cores that can handle floating point, integer and double precision as well as approximate transcendental operations.

Graphics Processing Units (GPU) were originally developed separately as special purpose processors to handle the computational load of real-time graphics displays through an algorithm known as rasterization. Incidentally, hardware architectures to perform fast rasterization are almost identical to the multicore designs. Combined with their cost benefit caused by the size of the gaming market, they quickly became a prominent architecture not
only for graphics but for general purpose parallel computations as well. Figure 3 above shows a block diagram of the architecture of a contemporary GPU.

For a program to benefit from GPU implementation, it must exhibit specific features that can be summarized as follows:

1. It is decomposable into many work items that can be processed independently.
2. The amount of parallel work should be larger than the number of available processors.
3. Processing of a single work item should avoid complicated control flow.
4. The parallel work items must comprise a significant part of the whole program, a principle also known as the Amdahl's law.

Such workloads are called "data parallel" and programs and algorithms often have to be redesigned and adapted to fulfill these conditions and so to fully benefit from the GPU.

### 2.1 Parallelization of SART

When implementing SART on CPU, the aim in general is to minimize the overall amount of numerical operations. This is best achieved by using a ray-based approach for both forward and back projection. Looking at the algorithm in Section 1, is SART a data parallel workload?

The first step is to realize that the forward projection, i.e. step a) from above can be processed entirely in parallel. Computations of each of the $\tilde{p}_i$ in one projection $P_p$ are independent from the rest of the $\tilde{p}_j$, since the volume values $v_i^{(k)}$ are only read. This means that there are about four million computation-heavy data parallel work items for a standard 2k x 2k detector. This obviously fulfills the conditions 1) and 2) given above. Condition 3) is not so obvious, but can also be achieved, although the details are out of the scope of this paper. It can thus be concluded that the forward projection phase is well parallelizable on GPU. However, for a significant benefit from a GPU implementation, also condition 4) should be fulfilled and so the back projection has to be effectively parallelized as well.

Though it is true for the CPU that a ray-based approach would work for the back projection as well, this is not the case for the GPU. The reason is that with a ray-based approach, multiple rays need to write to the same voxel. While this is no problem for a serial program, in a parallel program it requires synchronization between those rays, violating condition 1). A better approach is to turn the algorithm around and use a voxel-based approach. For each voxel, all rays from a given projection $P_p$ that contribute to this voxel are determined and their contribution computed according to Equation 2. In this way, the processing of each voxel is independent and can thus be run in parallel. For a serial program such an approach would be disadvantageous, since the process of determining the rays that influence each voxel is computationally demanding and the same computations are unnecessarily repeated over and over again. This is in general a recurring pattern in GPU programming: it is often worth to repeat a moderate number of arithmetic operations just to avoid communication and synchronization between work items.

Apart from general purpose computational cores, the GPU contains so called “texturing units”. These are specialized hardware units for performing very fast linear interpolation. These units can be used both in forward and back projection algorithms on the GPU to interpolate the pixel data. This in general results in smoother reconstructions. Such an
interpolation can also be performed on the CPU but without the specialized hardware units such operations would be simply too time consuming.

3. Experiments

3.1 Synthetic Data - Printed circuit board phantom

First we will inspect a synthetic data set, generated with Scorpius XLab. The phantom consists of a printed circuit board (PCB) with a ball grid array (BGA) and an IZFP logo. Additionally a crack has been positioned beneath the BGA. The data set was simulated in a CLARA geometry with a laminography angle of 45° and 200 projections of size $512^2$. The reconstruction was computed for a $128 \times 512 \times 512$ volume using three iterations of SART.

![Reconstruction of the IZFP PCB phantom](image1)

**Fig. 4.** Reconstruction of the IZFP PCB phantom. Front view of one slice through the reconstruction of the synthetic IZFP PCB phantom, left: CPU reconstruction, right: GPU reconstruction.

![Gray value comparison](image2)

**Fig. 5.** Gray value comparison between CPU and GPU synthetic reconstructions. Plot of gray values along the red line depicted in Figure 4 above. Blue curve represents CPU, green one GPU.

The difference in reconstruction quality between CPU and GPU implementations is negligible to the naked eye. If the gray value profile along a common line through the phantom is drawn for both reconstructions (Figure 5), the curves are still almost the same apart from some very slight variations in noise due to the different sampling strategies.
3.2. Measured Data - carbon fiber reinforced plastics

The second data set was acquired by the CLARA scanner of IZFP. A sheet of carbon fiber reinforced plastics (CFRP) with a complex fiber structure and small porosities was chosen as an example. The object was much larger than the detector, so we face the additional challenge of the so-called truncated data, which leads to artefacts in the reconstruction on the volume borders. 400 projections of size $2048^2$ were taken under a laminography angle of $30^\circ$. The reconstruction was computed for various volume sizes using three iterations of SART.

![Fig. 6. Reconstruction of the CFRP sheet. Front view of one slice through the reconstruction of the CFRP sheet, left: CPU reconstruction, right: GPU reconstruction. The slice shown was taken from the $128 \times 512 \times 512$ reconstruction. The defective porosity (orange circle) is clearly visible.](image)

![Fig. 7. Gray value comparison between CPU and GPU measured data reconstructions. Plot of gray values along the red line depicted in Figure 5 above. Blue curve represents CPU, green one the GPU.](image)

Compared to the results of the simulated noise-free data, the profiles of CPU and GPU reconstructions show a small difference in amplitude but the overall profile is almost identical. This difference is due to the influence of noise in the measured data, which is handled differently by voxel- and pixel-based reconstruction approaches, but does not detrimentally affect defect analysis as can be seen on the example of the clearly visible defective porosity.
4. Comparison of reconstruction times

For the comparison of the reconstruction times, a server-class Intel Xeon X5675 processor running at 3.07 GHz was used for the unoptimized single core CPU reconstruction. The optimized GPU algorithm was run on a mobile system equipped with a GeForce GT 730M and an Intel Core i7-4700MQ. For each reconstruction the total required time for three iterations was measured. The only exception is the full-sized CFRP dataset, where only one iteration has been timed. The reason is that the CPU reconstruction for that dataset would run for too long to provide useful data. From these times a speed up factor was computed describing the acceleration improvement of the GPU versus the CPU implementation. Table 1 shows the measured timings.

<table>
<thead>
<tr>
<th>Object</th>
<th>Iterations</th>
<th>Dimensions</th>
<th>CPU time</th>
<th>GPU time</th>
<th>Speed up factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>IZFP-PCB</td>
<td>3</td>
<td>128 x 512 x 512</td>
<td>1732s</td>
<td>128s</td>
<td>13.53x</td>
</tr>
<tr>
<td>CFRP</td>
<td>3</td>
<td>75 x 512 x 512</td>
<td>39023s</td>
<td>709s</td>
<td>55.04x</td>
</tr>
<tr>
<td>CFRP</td>
<td>3</td>
<td>150 x 1k x 1k</td>
<td>47677s</td>
<td>2344s</td>
<td>20.34x</td>
</tr>
<tr>
<td>CFRP</td>
<td>1</td>
<td>300 x 2k x 2k</td>
<td>28237s</td>
<td>3638s</td>
<td>7.76x</td>
</tr>
</tbody>
</table>

Table 1: Comparison of GPU and CPU reconstruction times. Depending on the situation, the GPU is anywhere between 8 and 55 times faster than an unoptimized single-core CPU implementation. Such speeds up bring the reconstruction down from hours to minutes, making CL an industrially viable appliance.

It can be seen that the GPU is anywhere between 7.76 and 55.04 times faster than the single CPU core. Let us first analyze the first and the last lines in the table. In these two reconstructions, the lateral size of the reconstructed dataset corresponds to the size of the projections (512 x 512 for the synthetic dataset and 2048 x 2048 for the measured dataset). In this configuration, the pixel-based and voxel-based back projection algorithms of CPU and GPU respectively perform equivalent operations (up to the linear interpolation mentioned in Section 2.1). So the speed up in these two cases corresponds to the net GPU advantage.

The second and third line are scenarios, where the reconstructed volume is smaller than the projections. In this case fewer rays are cast by the GPU voxel-based back projection algorithm. To avoid reconstruction artefacts, the GPU performs an additional filtering step in the pixel plane that is further accelerated by the hardware texturing units. Though this filtering step incurs some additional costs, it is in general cheaper that shooting the additional rays and so very large speed ups of 55x can be achieved. Moreover, from Figure 6 above, a slice from the 75 x 512 x 512 reconstruction, it can be clearly seen that the quality is equivalent even in the 55x speed up scenario.

As mentioned above, the unoptimized CPU algorithm was run on a server-class CPU, while the optimized GPU implementation used a mobile system. At first sight it may look like this would significantly influence the GPU results in a negative sense, as the GPU used is comparatively weaker than a GPU that might in reality be present in the server system. However, it is estimated that the real impact would be approximately a factor of 2, which does not play a major role given the numbers above.
5. Conclusion

For planar objects, which are not suitable for a computed tomography, computed laminography is the method of choice. It allows the inspection of industrially important products like fiber reinforced plastics and printed circuit boards. Using the innovative CLARA setup the mechanical requirements for a computed laminography system have been greatly reduced compared to standard laminographic scanners. The major obstacle to an industrial application of computed laminography so far has been the necessity of using iterative reconstruction methods to achieve high-quality reconstructions. These methods traditionally require a prohibitive amount of CPU computation time to be practically applicable.

In this work we have removed this obstacle by describing a GPU implementation of the SART algorithm that delivers high-quality reconstructions comparable to the CPU results, while being an order of magnitude faster than the CPU. This brings the reconstruction times down from hours to minutes. This result, together with the practical advantages of the CLARA scanner, demonstrates the feasibility of computed laminography for challenging industrially relevant inspections.

Part of this work was funded by the DFG grant "Parallel Iterative Methods with A Priori Information for Robust Computed Laminography of Low Contrast, Difficult-To-Measure Objects" [9].

References


8. DFG project "Parallel Iterative Methods with A Priori Information for Robust Computed Laminography of Low Contrast, Difficult-To-Measure Objects", www.imcl-sb.de