A real-time SAR image processing system for a millimetre wave radar NDT scanner

Christopher Schwäbig, Siying Wang, and Sabine Gütgemann

Fraunhofer-Institute for High Frequency Physics and Radar Techniques
FHR
Department Integrated Circuits and Sensor Systems
Fraunhoferstraße 20, 53343 Wachtberg, Germany

Abstract An ultra high resolution millimetre wave scanner for real-time SAR imaging is being developed at the Fraunhofer Institute for High Frequency Physics and Radar Techniques FHR. Highly integrated radar sensors with ultra wide bandwidth coupled with a new highly efficient SAR signal processing routine incorporate an illuminating scanner system for inline inspection of different materials and goods.

Keywords Extremely high frequency, SAR, real time image processing, CUDA®, NDT

1 Introduction

For various inspection tasks of different goods (for example food, 3D printed plastics) millimetre waves can be used. Millimetre waves depict the range from 30 GHz to 300 GHz of the electromagnetic spectrum. They are able to penetrate plastics, wood, glass and other materials with a low relative permittivity (ε_r). An advantage of millimetre waves in comparison to X-ray (which is also used for many inspection tasks) is the non-ionising radiation meaning that no special protection is necessary during the operation.

2 Current development status

The standalone millimetre wave imager ("SAMMI") [1] works with two antennas for transmitting and two antennas for receiving the

DOI: 10.58895/ksp/1000124383-17 erschienen in:
Forum Bildverarbeitung 2020
DOI: 10.5445/KSP/1000124383 | https://www.ksp.kit.edu/site/books/m/10.58895/ksp/1000124383/

electromagnetic wave. The transmitting antennas are placed underneath and the receiving antennas above a conveyor belt (see figure 2.1).

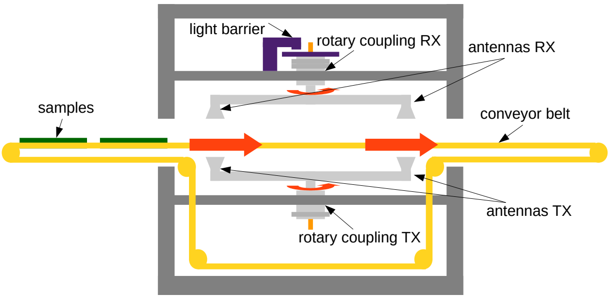

Figure 2.1: Mechanical concept of the current SAMMI version

A continuous wave signal at a frequency of 90 GHz is generated by a chain of several frequency mixers, filters and amplifiers. While the antenna pairs are moving along the conveyor belt (detected by a light barrier) an analogue digital converter samples the referenced raw data of the received signal and transfers it to the computer, where the transmission images are calculated and dynamically displayed in the user interface. These images are a 2D amplitude and a 2D phase image which represent the attenuation and the runtime of the electromagnetic wave within the object. SAMMI demonstrates the efficiency of radar technology and opens up development potential for further applications. To achieve a higher level of integration and above all, the possibility of the 3D imaging for nondestructive testing (NDT), an advanced system is developed as described in the following.

3 Development of a new SAMMI

As a result of these considerations a new system with a higher integration level, less mechanical expense and a frequency modulated radar is being developed (see figure 3.1). This enables the system to additionally provide depth information for real 3D imaging.

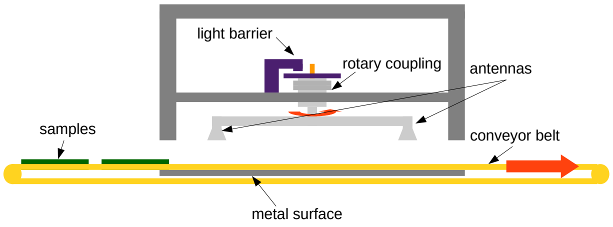

Figure 3.1: Mechanical concept of the SAR-SAMMI

The concept of this new SAMMI generation is based on synthetic-aperture radar (SAR) which is a retro-reflexive measurement method. For this approach two antennas above the conveyor belt are sufficient. In consequence the whole mechanical setup can be reduced so that mass and dimensions of the demonstrator decrease. Artefacts in the image due to the less ideal synchronicity of the antenna pairs do not exist. In contrast to the previous system the image processing algorithm becomes more complex.

4 Hardware signal processing

Radar sensors can be used for various applications for non-destructive testing [2], e. g. thickness measurement of multilayer samples [3] or determination of electromagnetic material properties. A FMCW radar based on a highly integrated SiGe radar chip [4] is used. The measurement principle is an indirect time-of-flight estimation of the difference between the transmitted and the reflected electromagnetic waves. For short range application the very high bandwidth of the radar chip allows ultra high resolution SAR imaging. During the measurement the received signals are collected, digitised and transferred to the computer (see figure 4.1).

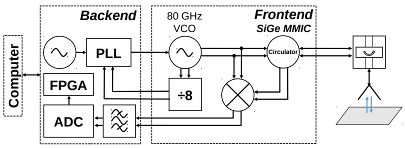

Figure 4.1: Hardware block diagram of the SAR-SAMMI

5 Image reconstruction signal processing chain and mathematical background

A highly efficient SAR algorithm is developed and implemented. During one single semi circular movement of the antenna 192 FMCW sweeps are recorded. The calculated results obtained from the data of one semi circular movement are stored within a temporary 3D array which is inserted in the final 3D output volume after the data of all sweeps have been processed.

5.1 Precalculations (only made once before the measurement)

The antenna positions during the circular antenna movement are presumed as constant so that a precalculation of all distances between the antenna positions and the voxels of the temporary 3D array can be done. Therefore calculation time for a real-time implementation decreases.

A precalculated mask for each sweep of the semi circle reveals whether or not the voxel of the temporary 3D volume can receive information of the current sweep. A look up table which is precalculated for each sweep of the semi circle, contains the distance for each voxel which has to be analysed (green in figure 5.1) in the sweeps spectrum.

5.2 Analysing the sweep of the semi circle

Each single sweep of the semi circle is filtered with a Hamming window in order to reduce the spectral leakage. FFT interpolation is used to achieve a higher resolution of the sweep signal in the frequency domain. If a voxel of the temporary 3D volume can receive

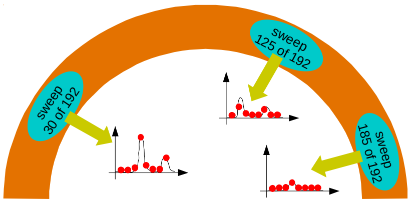

Figure 5.1: The sphere of influence (green) of different sweeps within the semi circle (orange)

information of the current sweep, the look up table and the nearest neighbour method are used to extract the image information from the sweeps spectrum. Based on the conventional backprojection algorithm [5] these values are weighted with a complex exponential function (see formula 5.1) dependent on their distance and are added into the temporary 3D volume (see figure 5.2).

$$s(m,\tau_n) = \texttt{fft}(s(f_k,\tau_n)) \cdot \exp\frac{+j4\pi f_1 \Delta R(m,\tau_n)}{c_0} \tag{5.1}$$

$s(m,\tau_n)$ describes the weighted value of *one* sweep for one voxel within the temporary 3D volume, dependent on the quantised range bin m within the sweeps spectrum and the time τ_n which depends on the number of the sweep. $s(f_k,\tau_n)$ describes the received quantised signal for one single sweep with k frequencies (f). $\Delta R(m,\tau_n)$ describes the quantised distance to the appropriated voxel. The frequency f_1 represents the start frequency of the FMCW down sweep or FMCW up sweep. For calculating the final value of one voxel of the temporary 3D volume, the values for $s(m,\tau_n)$ of all 192 sweeps have to be summed up.

5.3 Stepwise image build-up

After all sweeps of the semi circle have been processed, this temporary 3D volume is added into the final 3D volume. As a result of the movement of the conveyor belt the final 3D volume is shifted before the values of the temporary 3D volume are added. In order to create a x, y and z perspective of the final 3D volume, the amplitude values

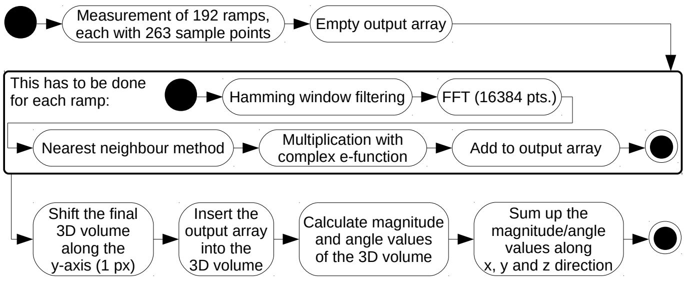
Measurement of 192 ramps, each with 263 sample points
Empty output array
This has to be done for each ramp:
Hamming window filtering
FFT (16384 pts.)
Nearest neighbour method
Multiplication with complex e-function
Add to output array
Shift the final 3D volume along the y-axis (1 px)
Insert the output array into the 3D volume
Calculate magnitude and angle values of the 3D volume
Sum up the magnitude/angle values along x, y and z direction

Figure 5.2: Software signal processing chain for each semi circle

are summed up along each axis. A whole image is created, based on multiple semi circles and the movement of the object (see figure 7.2). The speed of the conveyor belt depends on the chosen voxel size because the rotation speed of the antenna is assumed to be fixed.

6 Implementation

The implementation is divided into a CPU and a GPU part. The CPU part is separated into three threads and is responsible for GUI/visualisation, data capture (with hardware communication) and CPU-GPU data transfer. On the GPU the whole image processing chain is processed so that the array operations are calculated in parallel. Therefore the array operations are handled element by element and each array position is processed by its own thread within the GPU.

Before the actual measurement data evaluation can be processed, the precalculated look up tables are copied to the GPU. For each sweep (192 in total) of the semi circle two look up tables are computed. The first look up table illustrates whether a voxel within the temporary 3D volume can receive information from the current sweep or not. If a voxel can receive information from the current sweep the second look up table is used. This second look up table shows the distances (converted into FFT positions) for every voxel of the temporary 3D volume, which have to be analysed within the spectrum.

After these look up tables have been copied to the GPU the actual measurement process can be started. In this case a memory transfer (of precalculated data) is no longer necessary during the measurement which improves the execution time of the whole signal processing chain.

The different steps of the signal processing chain (see figure 5.2) are implemented in single functions (kernels). The batched CUDA® FFT routine (and respectively in general the CUDA® FFT) is called up by the host (the computer) and not by the device (the graphics card) itself. In connection with this, the other parts of the signal processing chain are implemented in the same way so that different CUDA® kernels represent the single steps of the signal processing chain. The sizes of the arrays (which are processed within the different functions) vary in size. In relation to this most of the different kernels (functions) vary in their thread and block configuration.

6.1 Detailed description of the single kernels within the signal processing chain

In the initial step the raw data of all 192 sweeps is copied to the GPU with help of the CUDA® kernel "cudaMemcpyAsync". Afterwards the raw data of all 192 sweeps is resorted from the hardware arrangement to a new arrangement so that the data can be directly processed with the batched FFT ("prepareDataForBatchedFFTWithWindowFilter"). In addition to this the raw data is filtered with a Hamming window. After the batched 1D FFT has been executed the temporary 3D volume is emptied (CUDA® kernel "cudaMemsetAsync"). This is necessary because this temporary 3D volume contains values of the previous semi circle. In the next steps this temporary 3D volume will be filled with processed data of the current semi circle.

The look up table and the nearest neighbour method are used to extract the necessary information from the corresponding spectrum ("nearestNeighbourMethodWithSum"). The extracted values of all 192 sweeps are summed up within the temporary 3D volume. Before the temporary 3D volume is inserted, the kernel "copyValues" creates a copy of the original final output 3D volume and the kernel "shiftValues" writes the values shifted by one element along the y axis in the original final output 3D volume. After the temporary 3D volume has been inserted (“insertMeasurement”) the absolute values of the complex final output 3D volume are calculated (“complex-ToAbs”). Based on the final output 3D volume with absolute values, three 2D images (one with the sums along the x-, one with the sums along the y-, and one with the sums along the z-axis) are calculated (“calculateXSum” / “calculateYSum” / “calculateZSum”). Within the last step these three 2D images are copied to the CPU (CUDA® kernel “cudaMemcpyAsync”).

6.2 Memory usage

Shared memory is useful when data stored within the global memory has to be accessed multiple times or data elements have to be shared between different threads in one block. In this implementation the implemented kernels are only responsible for a simple array operation and multiple access is not necessary. Therefore the use of shared memory by copying the data from the global memory into the shared memory is waived.

By using coalesced access to the global memory the execution time is much faster compared to uncoalesced access. In this case most of the kernels use a coalesced access to the memory in order to improve the speed of the algorithm. The kernels “prepare-DataForBatchedFFTWithWindowFilter” and “nearestNeighbourOptAllRampsWithSum” are much slower than the other kernels because their memory access is uncoalesced. The reason for this is that the kernel “prepareDataForBatchedFFTWithWindowFilter” resorts the input data (the algorithm has no bearing on the arrangement of this data) and the kernel “nearestNeighbourOptAllRampsWithSum” has to access various elements within the FFT spectrum (uncoalesced reading, but coalesced writing).

The implementation makes use of pinned memory in order to optimise the execution speed of the algorithm. In this case the recorded raw data is stored directly within the pinned memory. The same applies to the data which contain the three output images of the algorithm. In total, four arrays are declared as pinned memory (raw data input, output image x, output image y and output image z direction).

6.3 Further implementation details

In order to calculate the multiple 1D FFTs (192 in total) of one semi circle at once and not 192 single 1D FFTs this implementation makes use of the CUDA® batched 1D FFT. The advantage is that the execution routine of the FFT plan has only to be called once and not 192 times. The FFT plan consists the FFT settings and is created before the actual measurement is processed.

For the purpose of allowing further signalprocessing steps (for example phase unwrapping) the implementation takes advantage of CUDA® streams so that the semi circles can be processed in parallel by putting them into different GPU streams. To achieve this each signalprocessing chain for one stream has to be controlled by its own CPU thread. In this case each GPU stream is mapped with its separate plan of a batched 1D FFT.

The asynchronous memory transfer function "cudaMemcpy-Async" (host to device or device to host) is used so that each CPU thread can copy the data within its corresponding GPU stream. The same applies to the emptying process of the temporary 3D volume (before the steps of the signal processing are executed) with the function "cudaMemsetAsync".

Ideally the GPU streams are all processed in parallel. If the computing capacity of the operating graphics card is insufficient, parts of the processing run in sequence and in consequence the runtime increases.

7 Results

The created amplitude image (see figure 7.2, left) is based on simulated raw data of a 100 mm $\times$ 100 mm metal Siemens star (see figure 7.1).

In the simulated raw data the object is placed at a height of 30 mm which can be seen in both side views. The simulated raw data consists of 80 lines with 192 FMCW sweeps each (low resolution mode) and 160 lines with 192 FMCW sweeps each (high resolution mode). The raw data takes into account the semi circular movement path of the antenna. With the help of the algorithm this circular movement is corrected so that the measured object is depicted in the

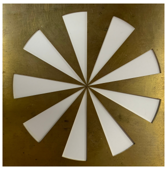
Figure 7.1: Photo of a Siemens star

correct shape and aspect ratio. More details concerning the object can be acquired by analysing the phase values which requires a 2D phase unwrapping algorithm [6].

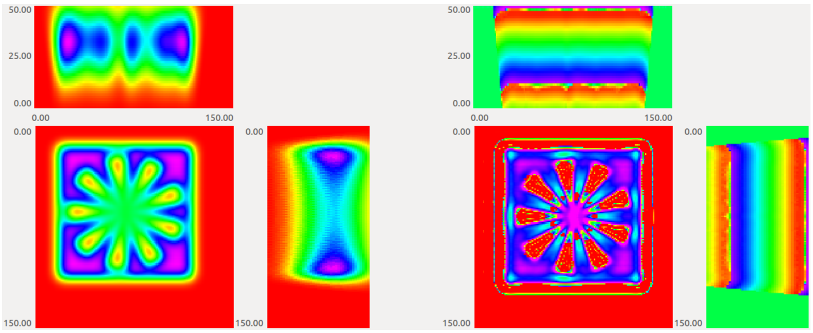

Figure 7.2: x, y and z view of the amplitude (left) and phase (right) 3D volume of a Siemens star

Because the Siemens star simulated here is infinitely thin, a top view (z axis) of the Siemens star can be generated without using a 2D phase unwrapping algorithm (see figure 7.2, right). The side views of the 3D phase image shows that the phase values pass through the whole phase spectrum.

7.1 Kernel runtimes

Table 1 shows the execution times of the different kernels of the signal processing chain for a low and high resolution compared on two graphics cards. It is obvious that the data preparation for the FFT, the FFT itself and the nearest neighbour method (with sum) consume most of the time (marked in red).

Table 1: Benchmark

Function / kernel	**NVIDIA® Quadro™P400 (2GB)**		**NVIDIA® Quadro RTX™6000 (24GB)**	
	Low resolution	*High resolution*	*Low resolution*	*High resolution*
Memcpy HtoD (async)	85.983 µs	n.p.[a]	45.056 µs	44.939 µs
prepareDataForBatchedFFT..	932.689 µs	n.p.[a]	38.496 µs	38.592 µs
Batched FFT	4183 µs	n.p.[a]	204.801 µs	202.113 µs
memset (Empty array)	24.256 µs	n.p.[a]	2.528 µs	21.28 µs
nearestNeighbourMethodWithSum	5340 µs	n.p.[a]	214.242 µs	2650.87 µs
copyValues	134.174 µs	n.p.[a]	5.696 µs	98.785 µs
shiftValues	398.362 µs	n.p.[a]	15.872 µs	273.826 µs
insertMeasurement	367.418 µs	n.p.[a]	9.28 µs	133.772 µs
complexToAbs	398.714 µs	n.p.[a]	13.408 µs	218.274 µs
calculateXSum	132.446 µs	n.p.[a]	17.568 µs	219.905 µs
Memcpy DtoH (async)	2.432 µs	n.p.[a]	1.472 µs	5.216 µs
calculateYSum	294.396 µs	n.p.[a]	29.568 µs	250.114 µs
Memcpy DtoH (async)	2.848 µs	n.p.[a]	1.44 µs	5.248 µs
calculateZSum	177.597 µs	n.p.[a]	10.272 µs	241.378 µs
Memcpy DtoH (async)	10.592 µs	n.p.[a]	5.216 µs	29.344 µs

[a] Not possible.

7.2 Memory and GPU utilisation

In the high resolution mode the required memory of the three dimensional arrays is around 15 times larger compared to the low resolution mode. The final three dimensional volume in the low resolution mode consists of 248897 values compared to 3624040 values in high resolution mode. The temporary three dimensional volume of one sweep in the low resolution mode consists of 100793 values compared to 1444800 values in high resolution mode.

In table 2 the memory usage and GPU utilisation during the measurement is shown. For the low resolution mode (2.5 mm), approx. 0.7 GB is used for the memory of the precalculated arrays and all the intermediate steps of the signal processing chain. Therefore on a small graphics card (for example NVIDIA® Quadro™P400, 2GB) 35 % of the memory is used. By using this graphics card the high resolution mode is not executable because of the higher memory requirements (10 GB).

The GPU utilisation is measured for four different input data rates. In the current implementation the use of CUDA® streams is not necessary because the input data rate of semi circles (10 Hz) is a lot lower than the maximum possible frame rate. Therefore the number of streams can be reduced to one.

Table 2: Memory usage and GPU utilisation

	NVIDIA® Quadro™P400 (2GB)		**NVIDIA® Quadro RTX™6000 (24GB)**	
	Low resolution	*High resolution*	*Low resolution*	*High resolution*
Memory signal processing	0.7 GB (35 %)	10 GB (500 %, n.p.[a])	0.7 GB (3 %)	10 GB (42 %)
Memory operating system	430 MB (22 %)	430 MB (22 %)	430 MB (2 %)	430 MB (2 %)
GPU utilisation	15 % (10 Hz) 30 % (20 Hz) 55 % (40 Hz) 90 % (67 Hz, max.)	n.p.[a] n.p.[a] n.p.[a] n.p.[a]	2 % (10 Hz) 3 % (20 Hz) 6 % (40 Hz) 60 % (350 Hz, max.)	6 % (10 Hz) 11 % (20 Hz) 22 % (40 Hz) 65 % (125 Hz, max.)

[a] Not possible.

Table 3 shows the execution speed of the different implementations for the low resolution and high resolution mode.

Table 3: Speed comparison

	Low resolution	*High resolution*
GNU Octave, CPU, one core	1.43 Hz	0.14 Hz
C Implement., CPU, one core	12 Hz	1.25 Hz
NVIDIA® Quadro™P400	67 Hz	n.p.[a]
NVIDIA® Quadro RTX™6000	350 Hz	125 Hz

[a] Not possible.

8 Conclusion

With the help of SAR it is possible to reduce the mechanical setup of the original standalone millimetre wave imager. By implementing the signal processing in CUDA® a real-time image generation is possible. Due to the fact that the achieved throughput is much higher than necessary, there is sufficient capacity for further signal processing steps (for example phase unwrapping).

References

1. A. Küter, C. Schwäbig, R. Brauns, S. Kose, and D. Nüßler, "A stand alone millimetre wave imaging scanner: System design and image anal-

ysis setup," *48th European Microwave Conference (EuMC), pp. 1505–1508*, 2018.

2. R. Herschel and S. Pawliczek, "3D millimeter wave screening of wind turbine blade segments," *15th European Radar Conference (EuRAD), pp. 115–117*, 2018.
3. S. Pawliczek, R. Herschel, and N. Pohl, "High precision surface reconstruction based on coherent near field synthetic aperture radar scans," *19th International Radar Symposium (IRS)*, 2018.
4. C. Bredendiek, K. Aufinger, and N. Pohl, "Full waveguide E- and W-band fundamental VCOs in SiGe:C technology for next generation FMCW radars sensors," *14th European Microwave Integrated Circuits Conference (EuMIC)*, 2019.
5. L. A. Gorham and L. J. Moore, "SAR image formation toolbox for MATLAB," *Algorithms for Synthetic Aperture Radar Imagery XVII*, 2010.
6. S. Pawliczek, R. Herschel, and N. Pohl, "3D millimeter wave screening for metallic surface defect detection," *16th European Radar Conference (EuRAD), pp. 113–116*, 2019.