GPU-BASED PARALLELIZED TIME-DOMAIN BACK-PROJECTION PROCESSING FOR AGILE SAR PLATFORMS

Othmar Frey
Gamma Remote Sensing
Switzerland
Earth Observation &
Remote Sensing, ETH Zurich
Email: frey@gamma-rs.ch

Charles L. Werner, Urs Wegmuller
Gamma Remote Sensing
Switzerland

***Abstract*—Agile SAR platforms such as an automobile require a flexible SAR processing scheme to account for nonlinear sensor trajectories during the synthetic aperture. In this contribution, a parallelized implementation of a time-domain back-projection SAR focusing algorithm based on NVIDIA's CUDA GPU computing framework is presented and discussed using a car-borne SAR data set. The processing performance is assessed using different hardware.**

In addition, a pre-processing scheme is described that allows for full 3-D motion compensation, yet staying conveniently in conventional slant-range / azimuth geometry of single-look complex SAR images.

***Index Terms*—Synthetic aperture radar (SAR), ground-based SAR system, SAR imaging, SAR interferometry, car-borne SAR, CARSAR , GPU, CUDA, Parallelization, Azimuth focusing, Nonlinear Sensor Trajectory**

I. Introduction

Agile synthetic aperture radar (SAR) platforms such as an automobile require a flexible SAR processing scheme to account for nonlinear sensor trajectories during the synthetic aperture. Recently, we have presented first results of such a car-borne SAR and InSAR experiment [1].

In this contribution, we present and assess a new parallelized implementation of a time-domain back-projection algorithm (TDBP) [2], [3] on a graphics processing unit (GPU) based on NVIDIA's Compute Unified Device Architecture (CUDA) application programming interface. We compare the processing performance of two implementations of the TDBP focusing algorithm, (1) an ANSI-C implementation on a CPU, and (2) a CUDA implementation of the same algorithm tested on three different classes of NVIDIA GPU devices.

In addition, a pre-processing scheme is described that allows for full 3-D motion compensation (3-D sensor coordinates and 3-D reconstruction grid) during TDBP processing, yet staying conveniently in conventional slant-range / azimuth geometry of single-look complex (SLC) SAR images.

TABLE I
GPRI-II GROUND BASED RADAR SYSTEM SPECIFICATIONS FOR SYNTHETIC APERTURE RADAR MODE.

Carrier frequency	17.2 GHz
Chirp bandwidth	200 MHz
Type	FMCW
Chirp length	0.001 s
Range 3dB beamwidth	18 deg
Azimuth 3dB beamwidth	16.9 deg
Ground speed	21 m/s
Interferometric baseline	0.25 m
Off-nadir angle	110 deg

The example data set used here to verify and assess the parallelized CUDA implementation of the time-domain back-projection algorithm was taken from a slightly curved road using a modified configuration of the GPRI-II ground-based radar [4], [5] mounted on the roof-top of a car (see [1] for more details). An overview of the most relevant system parameters is shown in Table I.

II. Methods

A. Range Compression

The car-borne SAR system used in this experiment (see [1]) is based on the linear FMCW-type GPRI-II radar [4], [5]. This radar operates in dechirp-on-receive mode, i.e. the received signal $s(t)$ is mixed with the reference signal. This transforms

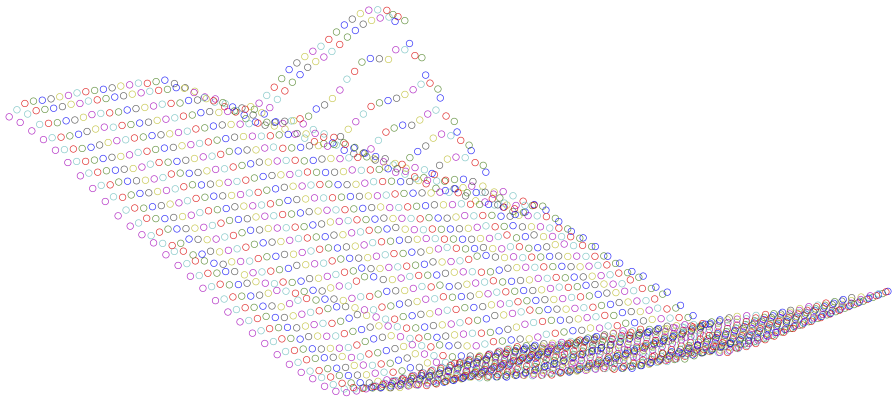

Fig. 1. Example of SLC (range/azimuth) geometry- type 3-D reconstruction grid obtained after DEM-resampling procedure.

the data to a deramped signal s_d of the form [6]:

$$s_d(t) = s^*(t)exp(j2\pi f_s t + j\pi\gamma t^2), \quad (1)$$

where f_s is the start frequency of the chirp and γ is the chirp rate. The phase of the resulting deramped signal is

$$\varphi_d(t) = (2\pi f_s t_n - \pi\gamma t_n^2) + 2\pi\gamma t_n t, \quad (2)$$

which can be directly related to range distance via a range FFT. t_n is the two-way time delay to a target n. For azimuth focusing of the data the residual video phase has to be compensated [7] and also the non-validity of the start-stop approximation must be taken into account [8].

B. Processing Steps to Prepare a DEM-based Image Reconstruction Grid in Range/Azimuth (SLC) Geometry

The following pre-processing scheme leads to a 3-D reconstruction grid with equispaced sampling in range and azimuth—i.e., a standard SLC data product—while still allowing for full 3-D motion compensation within TDBP azimuth focusing. It essentially involves resampling the DEM to a grid which is non-uniformly sampled in terms of its topocentric coordinates, however, which is sampled uniformly when considering its projection to the range-azimuth domain. The pre-processing sequence consists of the following steps:

1) Parameterize sensor track using orthogonal regression.
2) Estimate Doppler centroid frequency from range-compressed data.
3) Create a multilook geometry that serves to calculate the look-up table based SLC-type reconstruction grid more efficiently.
4) Create a geocoding lookup table based on a digital elevation model (DEM) and the parameterized sensor trajectory. The lookup table contains a floating point range and azimuth coordinate for each position in the DEM.
5) Invert the lookup table. The resulting inverted look-up table contains a floating point pixel coordinate of the DEM grid for each slant-range / azimuth coordinate.
6) Inverted look-up table is applied to 3-D DEM coordinates to obtain approximate topography information in the multi-looked geometry.
7) Upsampling of X, Y, and Z coordinates from MLI to SLC geometry.
8) Refinement of 3-D grid positions such that range distances match (to double precision) the nominal range distances.

See Fig. 1 for an example of such a resampled 3-D reconstruction grid with equispaced sampling in range and azimuth. The resolution is reduced for visualization purposes.

C. CUDA Implementation of TDBP Azimuth Focusing

The newly developed parallelized CUDA-based implementation of the TDBP algorithm on a GPU takes advantage of the single instruction-multiple data (SIMD) nature of the time-domain back-projection algorithm. The NVIDIA CUDA Fast Fourier Transform (FFT) library (cuFFT) [9] is used to implement the FFT-interpolation-based upsampling [10] of each echo. The SIMD-type operations during the actual back-

TABLE II
COMPARISON OF PROCESSING TIME OF CPU-BASED IMPLEMENTATION VS. 3 DIFFERENT CLASSES OF NVIDIA GPUS.

Processing unit	(V)RAM	No. of cores	Processing time	Speedup vs. CPU /	GT 650M
CPU: 2.6 GHz Intel Core i7	8 GB	4 (1 used)	1539 s	1	-
GPU: NVIDIA GeForce GT 650M	1 GB	384	315 s	4.9	1
GPU: NVIDIA GeForce GTX 660 Ti	2 GB	1344	79 s	19.5	4.0
GPU: NVIDIA Tesla K20c	4 GB	2496	51 s	30.2	6.2

projection operation exploit the optimized CUBLAS complex vector addition available in the NVIDIA CUDA Basic Linear Algebra Subroutines (cuBLAS) library [11], which is a GPU-accelerated implementation of the complete standard BLAS library.

In the current implementation the core processing sequence of the GPU-parallelized TDBP azimuth focusing consists of the following steps:

1) Read the sensor position and velocity data.
2) Read the coordinates of the image reconstruction grid.
3) Allocate memory on the GPU device for the SLC image block.
4) Allocate memory on the GPU device for the radar echo data, Doppler centroid values, and phase correction values per echo.
5) Start looping over radar echoes to perform back-projection.
    a) Update range- azimuth-varying Doppler centroid values to match the current radar echo in the loop.
    b) Update range varying phase correction values, which are updated for each azimuth echo.
    c) Copy the radar echo(s) to the device memory.
    d) FFT-based interpolation of the radar echo on the GPU device using the cuFFT library.
    e) Baseband to bandpass conversion of upsampled echo (introduce the time delay).
    f) Calculate the range distances between the current position and the coordinates of the reconstruction grid.
    g) Back-project the interpolated echo to an intermediate SLC image grid.
    h) Update the SLC image with the current back-projected data stored in the intermediate image grid.
6) End of back-projection loop.
7) Coherent baseband conversion of SLC image.
8) Copy the focused SLC image from the GPU device memory to the host memory.

III. RESULTS

In Table II a complete overview of the tested hardware and the corresponding processing times and speedup factors is given. The speedup factor obtained when comparing the two consumer-type NVIDIA cards GeForce GT 650M, found in laptop computers, and the NVIDIA GeForce GTX 660 Ti, designed for desktop computers, amounts to 315s / 79s = 4. The measured speedup scales roughly linearly with the 3.5 times larger number of available CUDA cores. Similarly, the speedup factor between GeForce GT 650M (Speedup : 315 / 51 = 6.2 times) scales approximately linearly with the 6.5 times larger number of CUDA cores (2496 vs 384) of the Tesla K20C and the GeForce GT 650M card, respectively.

Compared to the ANSI C CPU-based implementation of the time-domain back-projection algorithm a *maximal speedup of a factor of about 30 is achieved* with the Tesla K20C GPU.

Fig. 2 shows a visualisation of the processed SAR data taken from the CARSAR experiment [1]. The slightly curved sensor path is the lower-left part of the path highlighted in yellow color.

IV. DISCUSSION & CONCLUSION

A CUDA-based implementation of a time-domain back-projection (TDBP) algorithm for highly parallelized SAR azimuth focusing on GPUs was presented and assessed in terms of processing efficiency.

Already by using consumer-class GPUs found in laptops and desktop computers notable speedups of a factor of about 5 to 20 compared to the processing time on one CPU were obtained for the CUDA-implementation of the TDBP azimuth focusing algorithm. Using the high-performance Tesla K20c GPU card, a maximum speedup factor of about 30 was reached. In this case the processing time required for SAR focusing is in the same order of magnitude as the acquisition

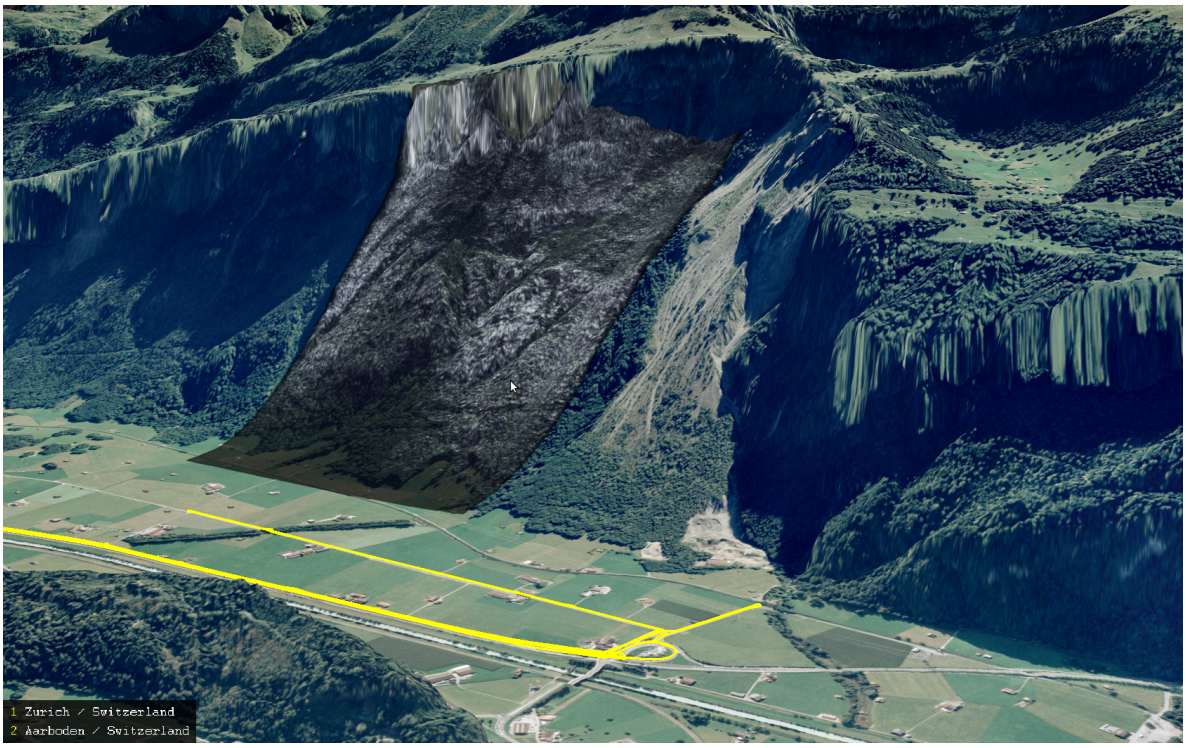

Fig. 2. Example SAR imagery of a slope of a valley taken from the car-borne interferometric SAR system. The SAR data is geocoded for 3-D visualization on top of orthophoto imagery and a high-resolution DEM (©swisstopo) of the area.

time for the car-borne SAR data, at hand, where the vehicle was driving at an average ground speed of 21 m/s.

GPU-based TDBP focusing is an attractive processing scheme for SAR data acquisitions involving curvilinear sensor trajectories. Examples include airborne SAR data acquired from highly nonlinear sensor trajectories [2] or focusing of car-borne SAR acquired along curvilinear roads for InSAR applications [1]. Other potential applications of the presented implementation include testing and reference processing for azimuth focusing of atypical SAR modes, such as e.g. geosynchronous SAR or blimp-borne SAR.

REFERENCES

[1] O. Frey, C. L. Werner, U. Wegmuller, A. Wiesmann, D. Henke, and C. Magnard, "A car-borne SAR and InSAR experiment," in *Proc. IEEE Int. Geosci. Remote Sens. Symp.*, 2013, pp. 93–96.

[2] O. Frey, C. Magnard, M. Rüegg, and E. Meier, "Focusing of airborne synthetic aperture radar data from highly nonlinear flight tracks," *IEEE Trans. Geosci. Remote Sens.*, vol. 47, no. 6, pp. 1844–1858, June 2009.

[3] O. Frey, E. Meier, and D. Nüesch, "Processing SAR data of rugged terrain by time-domain back-projection," in *SPIE Vol. 5980: SAR Image Analysis, Modeling, and Techniques X*, 2005.

[4] C. Werner, T. Strozzi, A. Wiesmann, and U. Wegmuller, "A real-aperture radar for ground-based differential interferometry," in *Proc. IEEE Int. Geosci. Remote Sens. Symp.*, vol. 3, July 2008, pp. 210–213.

[5] T. Strozzi, C. Werner, A. Wiesmann, and U. Wegmuller, "Topography mapping with a portable real-aperture radar interferometer," *IEEE Geosci. Remote Sens. Lett.*, vol. 9, no. 2, pp. 277–281, Mar. 2012.

[6] M. Soumekh, *Synthetic Aperture Radar Signal Processing: with MATLAB Algorithms*. John Wiley & Sons, 1999.

[7] W. G. Carrara, R. S. Goodman, and R. M. Majewski, *Spotlight Synthetic Aperture Radar: Signal Processing Algorithms*. Artech House Inc., 1995.

[8] A. Ribalta, "Time-domain reconstruction algorithms for FMCW-SAR," *IEEE Geoscience and Remote Sensing Letters*, vol. 8, no. 3, pp. 396–400, May 2011.

[9] "CUFFT library user's guide," NVIDIA Corporation, Tech. Rep., 2013, http://docs.nvidia.com/cuda/cufft.

[10] D. Fraser, "Interpolation by the FFT Revisited - an Experimental Investigation," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 37, no. 5, pp. 665–675, May 1989.

[11] "CUBLAS library user guide," NVIDIA Corporation, Tech. Rep., 2013, http://docs.nvidia.com/cuda/cublas.