Real-Time Processing of SAR Images for Linear and Non-Linear Tracks

Russel Que, Octavio Ponce, Rolf Scheiber, Andreas Reigber
Microwave and Radar Institute
German Aerospace Center (DLR)
Wessling, Germany
Email: russel.que@dlr.de

***Abstract*—A distributed real-time processing framework is used to implement a generic processor for linear and non-linear tracks. The focus is on the implementation of direct back-projection (DBP) and fast factorized back-projection (FFBP) algorithms in the framework. The software is described in detail on how it is tailored to achieve a real-time computation of a SAR image using multiprocessors, multicore CPUs and GPUs. Results are validated with airborne SAR data acquired by the DLR's F-SAR sensor.**

I. Introduction

Airborne platforms have been used ever since the inception of synthetic aperture radar (SAR). Although used mainly for linear tracks, a number of applications has seen a growing interest in non-linear tracks, particularly circular ones [1]-[4]. To process this new types of acquisition, the time-domain approach to SAR focusing has proven to be the most generic and flexible to implement [5],[6]. Direct back-projection (DBP) is seen as the most accurate algorithm but computationally intensive, in that, each image point is mapped to a point in the radar echo where the value is interpolated and added back to the image. Fast factorized back-projection (FFBP) is based on DBP, and it is able to reduce the number of computations, by using a more convenient reference system in polar coordinates while accomodating accurately azimuthal and topographic variations [1],[7].

As the algorithms improved, computing devices are also becoming more powerful. This in turn no longer limits the processing to powerful servers or to hours on personal workstations. As computing boards can be mounted on-board the platform, the possibilities open up to a myriad of real-time applications. There have been attempts to achieve real time capability in the past[8]-[11]. Here we present a new framework for on-board processing that takes inspiration from the one developed previously [12] for our multispectral, polarimetric and high-resolution airborne sensor F-SAR [13],[14]. It takes advantage of a collection of multiprocessor or multiple computers to implement a distributed computing architecture. With such a framework available, we implemented a generic processor using time domain approaches for airborne SAR acquired with arbitrary tracks, i.e., linear and non-linear.

II. Framework

1) Description: The computing framework [15] is a programmable streaming processor using its own script-like language to assign or run function modules on different nodes. After creation of modules, the ethernet interconnections between them are established, depending on the input/output requirements of each. In turn, the modules which live in their own thread run continuously, while ingesting input data from the ethernet streams and then sending out the results to one or multiple destination modules. Fig. 1 illustrates a workflow example.

2) Workflow: Processing on the fly can be a daunting task, especially when using distributed computing with very high data rate, as it requires careful planning of resources like network capacity, memory, hard disk, multi-core CPUs, GPUs and others. The complexity of this problem can be solved by parallelization. Division of the application can first be made across different nodes, where each node accepts data streams, performs a task or algorithm then passes the output to another. The framework does not have a star-based topology but is rather, fully-connected, thereby saving network traffic by allowing the data to be passed directly to the next module anywhere in the network.

The second level in which parallelization can be employed is in dividing the task further by using multi-core / multi-threading programming or by offloading the computational task to a graphical processing unit (GPU). Fig. 2 illustrates the workflow for the generic processor we have developed for arbitrary tracks. In this example, two nodes are used in case of single-channel SAR acquisitions. However, additional nodes can be added when more channels are required. The first node would be the control node where the operator manages the radar and processing system. Therein lies the execution of the workflow script that setups the processing modules in Node A and the rest. It contains as well the graphical user interface (GUI) to view and verify results from the modules. Node A, in particular, has three modules running in it. But since *PrepareRadarParameters* and *PrepareNavigation* are not computationally intensive, the *RangeCompression* module is also run in the same node. The *TimeDomainFocusingAlgorithm* has Node B all to itself.

III. Streaming Architecture of the Focusing Algorithm Module

This module is the one responsible for focusing the compressed radar echoes into a SAR image. In order to have a

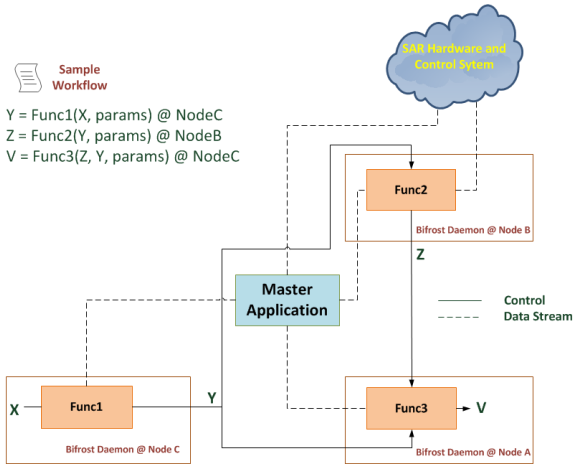

Fig. 1. Workflow setup. A workflow script describes the interconnection of a master application to the function modules it spawns in remote nodes.

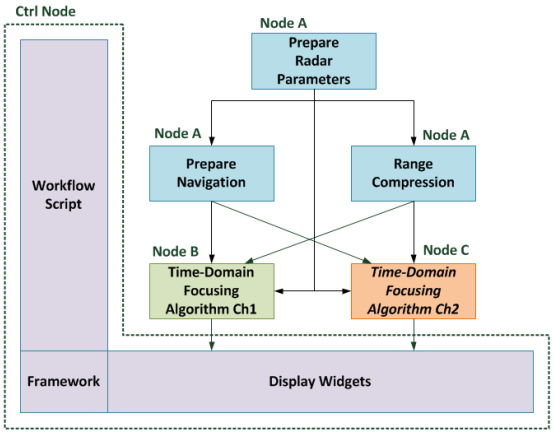

Fig. 2. Framework workflow of the generic processor for all types of tracks.

generic processor for all types of sensor tracks, be they linear, circular or on any arbitrary track, the time-domain method would be the most flexible approach for all those cases. In particular, the DBP and FFBP algotrithms are implemented in this processor.

The next crucial design step tackles the implementation constraints and opportunities presented by the target machine platform. Principally, the goal is how to design the most efficient way to handle tens of gigabytes of streaming range compressed data with only a standard memory, currently 16GB, and 1Gbps ethernet connection. Optimizing the transfer of data should be prioritized, therefore, the ingestion of data into data containers in memory is programmed to be non-stop and independent of the kernel algorithm thread. When the algorithm finished processing one data container, it simply takes the next one from the queue without having to wait or activate the TCP transfer. Since the memory is finite and the algorithm itself needs a big chunk of free space, the total memory size of the data containers in the queue is limited to a few hundred megabytes.

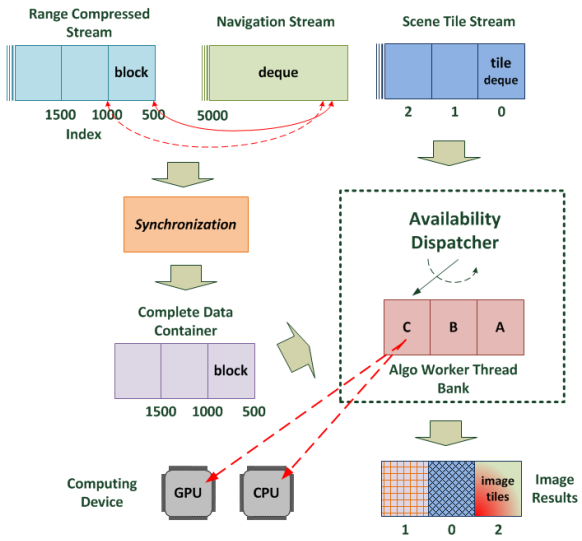

Fig. 3. Streaming architecture and partitioning inside the focusing algorithm module.

Aside from the range-compressed data, the navigation data is also streaming in real time (see Fig. 3). Hence, in each of the data containers, the navigation data is filtered and synchronized with the SAR data. Upon alignment, the data container is deemed complete and ready to be focused and projected onto the scene. The scene however, as a geographic coordinate grid, is also a streaming data from the navigation module. It is automatically partitioned in a tiled arrangement by computing if a particular tile is illuminated within the ground footprint of the antenna. The idea of processing by tiles is to avoid cluttering the memory with scenes that have been already completely projected and scenes that are yet to be relevant or 'seen' by the current and active data containers. Once a data container is fully projected onto all the relevant scene tiles, it is discarded from the memory. Hence, at any given time, the memory space is only occupied by active data containers and active scene tiles, thereby reserving the rest of the memory space to the algorithm thread.

As depicted in Fig. 3, the projection of the data container to the scene tile is scheduled by a dispatcher that checks if there is a worker thread available. The size of the worker bank can be chosen to be the number of CPU cores or less if the algorithm kernel is also already heavily parallelized/multi-threaded. To take advantage of the powerful computing devices like a GPU or Custom FPGA that can be attached to the machine node, the worker thread has the possibility to offload the computation to those. In the implementation of the framework, there is an option to choose the computing device of the worker thread, namely, CPU or GPU.

IV. Time-Domain Focusing Algorithms

A. Direct Back-Projection

The direct back-projection is an exact method where a pixel or position in space is traced to an index in a particular

range-compressed echo. The index is computed with respect to the distance travelled from the transmit antenna, to the pixel target and back to the receive antenna. It is however, not a whole integer, and hence requires interpolation of the range-compressed data. The values interpolated from each of the azimuth range lines are then added coherently to produce a focused image.

It is this pixel to range-compressed data mapping approach that enables a highly resolved image-focusing regardless of the track geometry. However, to obtain such a well-focused SAR image, a simple linear interpolation will not suffice. It can still be used, nonetheless, but only after performing a FFT zero padding to first produce an up-sampled range-compressed data. This approach is the one implemented in this paper for the DBP, both in CPU and GPU. In some cases, like FFBP in the next section, it is preferable and more efficient to avoid the FFT zero padding and instead use a better interpolation technique like sinc, cubic or others.

B. Fast Factorized Back-Projection

Overall, DBP, however exact and flexible it may be, is still an extremely time-consuming approach. The FFBP was introduced in [7] and it is used as a computationally efficient solution to the DBP. It reduces the number of computations significantly from a cubic M^3 to a logarithmic $M^2 * log_2(M)$ behaviour, with respect to DBP. This is achieved by splitting the synthetic aperture in smaller subapertures and also by the resulting subaperture images in polar coordinates, which reduce significantly the sampling requirements in the along-track direction. Originally, this algorithm has been developed for linear tracks, but a modified version for non-linear tracks has been implemented in [1].

In order to tailor the FFBP for real time applications, the original version has been modified [10]. Fig. 4 shows the divide-and-conquer approach wherein the dataset is partitioned into several angular apertures denoted by the *t* progression, and how those are combined with polar-to-polar interpolations over several stages *k* to produce new subaperture images. In the final stage, the resulting subaperture image is projected onto a geocoded grid through polar-to-cartesian (P2C) interpolations. As illustrated in Fig. 4, the processing flow is stopped when the desired resolution is achieved at a given time and stage. One additional advantage of the FFBP for real-time applications is that the processing of the range compressed data can start at the very beginning of the acquisition, i.e., without waiting for big blocks of raw data to be preprocessed as seen in frequency-based algorithms. The process of recomputing the dataset, which involves angular realigning and interpolations, e.g., P2P and P2C, over several stages, do require some computational effort. But in total, it is far less time consuming than the intensive effort needed when projecting range line by range line as in DBP.

V. First Results

Shown on Fig. 5 is the result of a linear flight track processed on a GPU using DBP. The three yellow slanted lines

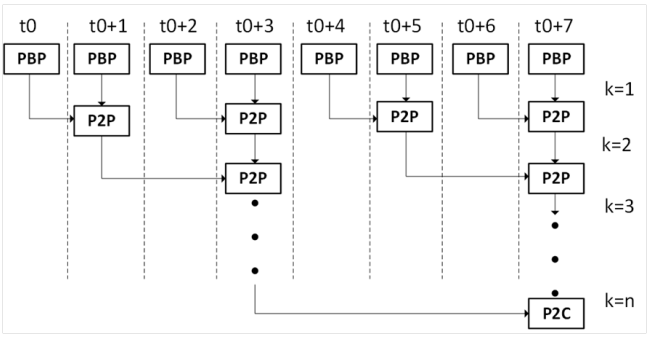

Fig. 4. Fast factorized back-projection (FFBP) dependencies over time for the real-time processor. Depending on the desired resolution, the processing chain can be stopped at a given time $t_0 + i$ and stage k. PBP indicates the polar back-projection, while P2P and P2C refers to polar-to-polar interpolations and polar-to-cartesian interpolation.

from right to left show the ground track of the near-range, main beam track, and the far-range. In this figure the near-range is selected to be the beginning of the echo, hence there are black or invalid regions comprising some of the tiles. The far-range is configured to be less than the length of the radar echo. Both near and far-range are configurable, and in between those two ground tracks will the scene tiles be automatically selected, as evidently shown on the figure being ringed by blue lines. Therefore, tiles that are activated essentially follow the course of the sensor's illuminated ground track be it curved, straight or stays constant in case of circular path.

The presented example took about 6 mins to process a 2m x 2m grid resolution, 4°aperture and 4 looks using the DBP algorithm on GPU to accelerate computation, which otherwise, would take hours in a conventional CPU. With this specific algorithm, cutting the grid resolution by a factor, on a particular dimension, would cut as well the time by almost the same factor. So by selecting 4m x 4m, in case of quicklook mode, it would see the processing time reduced by a total factor of 4, down to 1.5 to 2 mins. Moreover, by using a block-FFT presumming, it can further reduce the focusing time by a factor equivalent to the downsampling in azimuth.

An example of a curved/circular data take is shown in Fig. 6. The blue line represents the beam track on ground. As the image shows, the tiles are correctly selected even for non-linear tracks. Also, the adaptation of the algorithm works correctly although the scene illumination is moving in an arbitrary direction. The radiometric calibration is performed for this case by computing the illumination history of the ground together with the antenna beam profile and navigation data.

VI. Conclusions and Future Work

We have shown so far that the new framework can be used to process close to real-time a linear track in spite of using a very inefficient algorithm. Although the initial test algorithm used is DBP, it was made to work fast enough due to the use of GPU. The FFBP re-implementation in C++ and its integration to the framework is still in preliminary stages. Once finished,

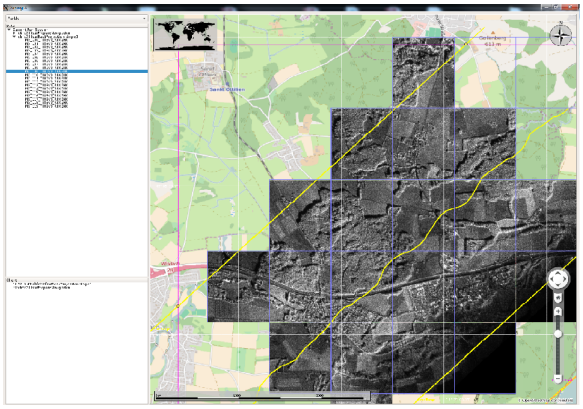

Fig. 5. Linear track; SAR image result tiles using DBP on GPU(Nvidia Tesla C2070). Displayed on KDE Marble map widget with Openstreetmap. Grid resolution 2 m x 2 m, 4 multilooks, 4° azimuthal aperture and no presumming.

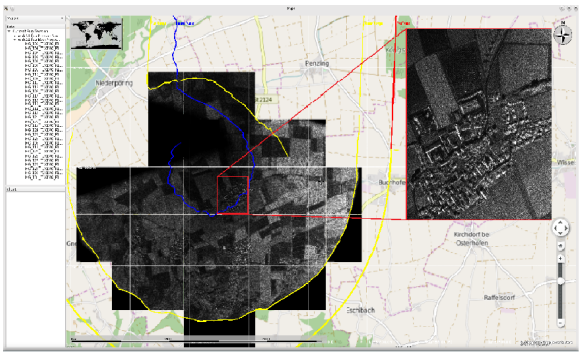

Fig. 6. Curved or circular track; Grid resolution 1.5 m x 1.5 m, single look, 4° azimuthal aperture and no presumming.

we expect that the speedup gained from it to be considerable making it truly real-time. Future work will also entail creating post-processing modules for real-time applications involving multiple channels like polarimetry, GMTI and interferometry.

REFERENCES

[1] Ponce, O., Prats, P., Pinheiro, M., Rodriguez-Cassola, M., Scheiber, R., Reigber, A., Moreira, A., *Fully-Polarimetric High-Resolution 3-D Imaging with Circular SAR at L-Band*, IEEE Trans. Geosci. Remote Sensing, Vol. 52, 2014.

[2] P.-O. Froelind, A. Gustavsson, M. Lundberg, and L. M. H. Ulander, *Circular-aperture VHF-band synthetic aperture radar for detection of vehicles in forest concealment"*, IEEE Trans. Geosci. Remote Sens., vol. 50, no. 4, pp. 13291339, Apr. 2012.

[3] L. J. Moore and L. C. Potter, *Three-dimensional resolution for circular synthetic aperture radar"*, Proc. Algorithms Synthetic Aperture Radar Imagery XIV, vol. 6568, SPIE Defense Security, Apr. 2007, p. 656 804

[4] E. Collin and H. Cantalloube, *Assessment of physical limitations of high resolution on targets at X-band from circular SAR experiments*, presented at the EUSAR, Jun. 2008 pp. 14.

[5] M. Soumekh, *Synthetic Aperture Radar Signal Processing:With MATLAB Algorithms*, Hoboken, NJ, USA: Wiley, 1999.

[6] M. Albuquerque, P. Prats, and R. Scheiber, *"Applications of time-domain back-projection sar processing in the airborne case"*, in Synthetic Aperture Radar (EUSAR), 2008 7th European Conference on, June 2008, pp. 1-4

[7] Lars M.H. Ulander, *Synthetic-Aperture Radar Processing Using Fast Factorized Back-Projection*, IEEE Transactions on Aerospace and Electronic Systems , vol. 39 , no. 3 , pp.760 -776 , 2003

[8] Cantalloube, H.M.J., *Real-time Airborne SAR Imaging. Motion compensation and Autofocus issues*, EUSAR Conference 2012, Munich, Germany, 2012.

[9] Simon-Klar, C., Friebe, L., Kloos, H., Lieske, H., Hinrichs, W., Pirsch, P., *A Multi DSP Board for Real Time SAR Processing using the HiPAR-DSP 16*, IEEE,, 2002.

[10] Lidberg, C., Olin, J., *Optimization of Fast Factorized Backprojection execution performance*, Chalmers University of Technology, Sweden, 2012.

[11] Hast, A., Johansson, L., *Fast Factorized Back-Projection in an FPGA*, Halmstad University, Sweden, 2006.

[12] Andres, C.; Keil, T.; Herrmann, R.; Scheiber, R., *A multiprocessing framework for SAR image processing*, Geoscience and Remote Sensing Symposium, IGARSS 2007.

[13] Reigber, A., Scheiber, R., Jäger, M., Prats, P., Hajnsek, I., Jagdhuber, T., Papathanassiou, K., Nannini, M., Aguilera, E., Baumgartner, S.V., Horn, R., Nottensteiner, A., Moreira, A., *Very-High-Resolution Airborne Synthetic Aperture Radar Imaging: Signal Processing and Applications*, Proceedings of the IEEE, vol. 101, no. 3, pp.759-783, 2013

[14] Jäger, M., Pinheiro, M., Ponce, O., Reigber, A., Scheiber, R., *A Survey of Novel Airborne SAR Signal Processing Techniques and Applications for DLR's F-SAR Sensor*, International Radar Symposium (ISSN: 0885-8985), 2015

[15] Que, R., Ponce, O., Baumgartner, S.V., Scheiber, R., *Multi-mode Real-Time SAR On-Board Processing*, 11th European Conference on Synthetic Aperture Radar (EUSAR).