Progress In Electromagnetics Research, Vol. 133, 259–283, 2013

FAST GPU-BASED INTERPOLATION FOR SAR BACK-PROJECTION

A. Capozzoli*, C. Curcio, and A. Liseno

Dipartimento di Ingegneria Biomedica, Elettronica e delle Telecomunicazioni, Università di Napoli Federico II, Via Claudio 21, I-80125 Napoli, Italy

Abstract—We introduce and discuss a parallel SAR backprojection algorithm using a Non-Uniform FFT (NUFFT) routine implemented on a GPU in CUDA language.

The details of a convenient GPU implementation of the NUFFT-based SAR backprojection algorithm, amenable to further generalizations to a multi-GPU architecture, are also given.

The performance of the approach is analyzed in terms of accuracy and computational speed by comparisons to a "standard", parallel version of the backprojection algorithm exploiting FFT + interpolation instead of the NUFFT. Different interpolators have been considered for the latter processing scheme. The NUFFT-based backprojection has proven significantly more accurate than all the compared approach, with a computing time of the same order. An analysis of the computational burden of all the different steps involved in both the considered approaches (i.e., standard and NUFFT backprojections) has been also reported.

Experimental results against the Air Force Research Laboratory (AFRL) airborne data delivered under the "challenge problem for SAR-based Ground Moving Target Identification (GMTI) in urban environments" and collected over circular flight paths are also shown.

1. INTRODUCTION

The backprojection algorithm [1, 2] is a frequently used technique in Synthetic Aperture Radar (SAR) image formation. Assuming that the data acquired by the SAR sensor are available over a uniform grid in the frequency domain, SAR backprojection consists of two-steps.

Received 19 July 2012, Accepted 17 September 2012, Scheduled 22 October 2012

* Corresponding author: Amedeo Capozzoli (a.capozzoli@unina.it).

The first step amounts to transform the frequency domain data by a Fast Fourier Transform (FFT) to a uniform grid of the time domain, while the second step regards interpolating the data from the uniform grid dictated by the FFT to the non-uniform grid dictated by the discretization of the on-ground scenario.

The backprojection algorithm shows two main advantages if compared to other approaches [1]. First, it does not involve approximations of the relevant Green's function. Second, it enables to form SAR images as the data are acquired, pulse by pulse, by integrating newly obtained information into the SAR image as it becomes available. The main drawback of SAR backprojection is the computational complexity, which grows, in a sequential algorithm, as $P(M) + O(M^2 \log M)$ for an $M \times M$ image, where the polynomial dependence $P(M)$ is due to the required interpolation stage, while the $O(M^2 \log M)$ dependence is due to the involved FFT step [3, 4]. The possibility of using FFTs is thus appealing from the computational viewpoint. Indeed, for a fixed M and depending on the interpolation scheme, the dominant term can become $P(M)$ or $O(M^2 \log M)$. Thus, the choice of the interpolator is critical to obtain a favourable trade-off between accuracy and processing speed, since accurate but slow interpolators can make the approach for very large data sets even impractical [2]. In other words, proper interpolators can enable the algorithm to have the same asymptotic computational complexity of a standard FFT.

To reduce the computational burden, different methods have been proposed. While accepting some sacrifice in image quality, they rely on FFT stages performed in polar coordinates [5] or based on hierarchical decompositions of the backprojection operator [6, 7] to achieve an $O(M^2 \log M)$ complexity.

Approaches based on the use of Non-Uniform FFTs (NUFFTs) [8, 9] have recently gained interest in SAR imaging to enable the efficient computation ($O(M^2 \log M)$ complexity) of Non-Uniform Discrete Fourier Transforms (NUDFTs) with unequally spaced data or results [10, 11]. NUFFT-based backprojection represents an interesting alternative to other backprojection approaches to efficiently nest in a single stage the FFT and an effective and precise interpolation, and to form very accurate SAR images at a reasonable computational cost.

We stress that accurately and quickly processing SAR data by the use of effective algorithms and performing hardware has aroused interest since more than two decades. This happened since when the FFT algorithm was used, for the first times, to quicken the computationally burdened ω-k processing [12–14], the use of travelling sampling interpolation techniques for bandlimited functions were

firstly experimented [14] and computer parallel codes were firstly developed to accelerate the computations [15]. Currently, it arouses interest in all those applications in which obtaining precise images of large areas [16] or accurately extracting geometrical [17–19] or physical parameters [20, 21] is required. For example, in SAR interferometry, guaranteeing high coherence values to preserve the accuracy of the interferometric phase and to obtain meaningful Digital Elevation Maps (DEMs) is crucial [22]. It is no accident that right in this framework different interpolators have been proposed and compared in terms of accuracy [17–19].

It should be also noticed that SAR backprojection lends naturally itself to parallelization, which should naturally lead to the choice of parallel hardware for its implementation. Graphics Processing Units (GPUs), which provide platforms for parallel computing with very competitive "flops per dollar" ratios [23–28], represent an excellent opportunity in this framework. The use of GPUs is spreading over the SAR community [10, 11, 29–34] whose first attempts in this context date even back to the times when extensions of the ANSI C simplifying the programming of graphic cards (as CUDA — Compute Unified Device Architecture — or OpenCL) [35] were not yet available [36].

Purpose of this paper is to introduce, for the very first time, and discuss a new parallel SAR backprojection algorithm using a NUFFT routine implemented on GPU. The details of a convenient GPU implementation of the NUFFT-based SAR backprojection algorithm, amenable to further generalizations to a multi-GPU architecture [34], are also given. We compare, under the new light of parallel programming on GPUs, SAR backprojection algorithms based on the most common interpolators proposed by the SAR community and the proposed SAR backprojection approach based on the use of the NUFFT [10, 11, 37]. The comparison is performed in terms of accuracy and processing speed, by highlighting the computational performance of each individual step of the involved routines. The main results are highlighted in detail in the Conclusions and Future Developments Section. The extent of the comparison against other pre-existing techniques can result also very useful to the Reader who is then oriented to the choice of interest depending on the needs, again in terms of accuracy and processing speed. The very convenient performance of the NUFFT-based approach is confirmed by results obtained against data provided by the Air Force Research Laboratory (AFRL) under the "challenge problem for SAR-based Ground Moving Target Identification (GMTI) in urban environments" [38] and collected by the airborne sensor of the AFRL under circular flight paths.

The paper is organized as follows.

In Section 2, the SAR signal model is shortly recalled. In Section 3, the backprojection algorithm is described by paying particular attention to the FFT+interpolation stage and by discussing how the latter can be implemented by a Non Equispaced Results (NER) NUFFT [8, 9]. Section 4 points out the computational complexity of the FFT+interpolation stage, by stressing the advantage of using a NER-NUFFT. Furthermore, the interpolation kernels considered in this paper and used for the comparison with the NUFFT-based SAR backprojection are detailed and the NER-NUFFT algorithm is briefly recalled. Section 5 is devoted to shortly illustrate the GPU implementations of the confronted algorithms while in Section 6 we present numerical results concerning their comparison in terms of accuracy and processing speed. Section 7 contains the experimental results on the AFRL-GMTI data which illustrate the outperforming features of the NUFFT in terms of achievable accuracy only, being the processing speed analyzed in Section 6. Finally, in Section 8, conclusions are gathered and ongoing developments are illustrated.

2. SAR SIGNAL MODEL

Let us consider the monostatic SAR acquisition geometry of Fig. 1.

The SAR sensor travels along a flight path such that $(x(\tau), y(\tau), z(\tau))$ represents the trajectory of the Transmitting (Tx)/Receiving (Rx) antenna phase center, where τ stands for the "slow-time" [2]. We consider a scene populated by generic stationary targets (i.e., their locations do not depend on τ), residing at the

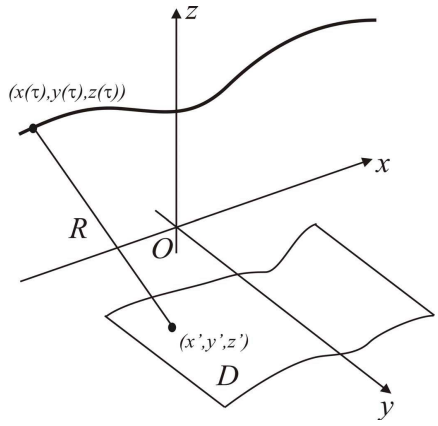

Figure 1. Geometry of the problem.

positions (x', y', z'), and we assume that the target reflecting features do not depend neither on the frequency f nor on the aspect angle, which is related to the reciprocal positions of the sensor and the scatterers themselves. As the antenna phase center moves along the flight path, it transmits pulses (chirp pulses, typically) in the direction of the scene at regularly spaced time instants τ_n.

Following the "start-stop" approximation, at a given τ_n, the output of the receiver is a function of the "fast-time" t [1, 2] denoted by $s(t, \tau_n)$ and known as "raw-data". Its frequency domain expression is

$$S(f, \tau_n) = H(f) \iint_D A\left(x', y'\right) e^{-j4\pi f \Delta R(x', y'; \tau_n)/c} dx' dy', \tag{1}$$

where the integration is extended to the investigated scenario D,

$$\Delta R\left(x', y'; \tau_n\right) = \sqrt{\left(x(\tau_n) - x'\right)^2 + \left(y(\tau_n) - y'\right)^2 + \left(z(\tau_n) - z'\left(x', y'\right)\right)^2} - \sqrt{x^2(\tau_n) + y^2(\tau_n) + z^2(\tau_n)}, \tag{2}$$

the scatterers are assumed to be located on the surface of equation $z' = z'(x', y')$, A is the "scattering amplitude" of the generic target at (x', y', z') [2], $H(f)$ is the spectrum of the illuminating pulse and c is the speed of light. In Eq. (1), it has been assumed, as in common practice, that a scatterer at the scene origin O will have zero phase for every f and τ_n.

If a DEM is available, then the actual target locations can be accounted for. Since our purpose is to analyze different processing schemes in terms of accuracy and computational performance, we assume no further a priori information so that in the next Sections we set $z' = 0$.

3. THE BACKPROJECTION ALGORITHM

On defining an (x'_m, y'_m) grid for D, an approximation to A can be expressed as [10, 11]

$$A\left(x'_m, y'_m\right) = \sum_n Q_{mn}\left(t_{mn}^{NU}\right) \tag{3}$$

where

$$Q_{mn}\left(t_{mn}^{NU}\right) = \int_{f_{\min}}^{f_{\max}} \frac{S(f, \tau_n)}{H(f)} e^{j2\pi f t_{mn}^{NU}} f df \tag{4}$$

is the "filtered projection", $f_{\min}$ and $f_{\max}$ are the minimum and maximum frequencies of S, respectively, and the t_{mn}^{NU}'s are the fast-time instants given by

$$t_{mn}^{NU} = 4\pi \frac{\Delta R\left(x'_m, y'_m; \tau_n\right)}{c}. \tag{5}$$

It should be noticed that dividing the acquired signal $S(f, \tau_n)$ by $H(f)$ in Eq. (4) corresponds to range-compression or, in particular, to standard matched filtering when the chirp signal is used as illuminating pulse [39]. It should be also stressed that, since the fast-time instants t_{mn}^{NU} and the grid coordinates (x'_m, y'_m) are non-linearly related by Eq. (5), then the t_{mn}^{NU}'s are non-uniformly spaced, even for a uniform Cartesian grid (x'_m, y'_m).

After having computed $S(f, \tau_n)$ from the acquired raw-data by FFT operations [2], the evaluation of the filtered projections can be performed following two different ways.

(i) **"Standard backprojection"**, henceforth **"Algorithm A"**. A first possibility is to use an FFT routine, which however provides the Q_{mn}'s at uniformly spaced fast-time instants t_{mn}^{U}. Consequently, an interpolation step is also required to evaluate the filtered projections from the uniform grid t_{mn}^{U} (provided by the FFT) to the non-uniform grid t_{mn}^{NU} of interest.

(ii) **"NUFFT-based backprojection"**, henceforth **"Algorithm B"**. Alternatively, a 1D NUFFT routine of NER type can be used to achieve the values of the filtered projections directly at the non-uniform sampling points of interest [10, 11].

4. COMPUTATIONAL COMPLEXITY AND INTERPOLATION

For a sequential approach and for an image of size $N \times M$ (azimuth $\times$ range), the computational complexity of step #1 is $O(NM \log M)$. On the other side, as stressed in the Introduction, the complexity of steps #2 and #3 is, apart from oversampling factors [8, 9], $O(NM \log M) + P(N, M)$. It should be noticed that, if the evaluation of $Q_{mn}\left(t_{mn}^{NU}\right)$ involves a number of, say, $T \ll M$ samples of $Q_{mn}\left(t_{mn}^{U}\right)$ ("traveling" sampling interpolation [40]), then $P(N, M) \simeq NTM$, so that the overall complexity of the backprojection algorithm becomes dominated by the term $O\left(NM \log M\right)$ for large images. Different polynomial interpolation kernels or procedures to accelerate the convergence speed of the Nyquist series by proper window functions have been proposed and considered in the SAR literature [17–19, 41, 42]. Conversely, the Nyquist series used without convergence acceleration entails $T \simeq M$, so that $P(N, M) \simeq NM^2$ dominates the computational complexity [32].

4.1. Interpolation Kernels

The interpolation kernels herein considered for Algorithm A give rise to traveling sampling interpolators and are the following (assuming

unit uniform sample grid distance):

- **Nearest neighbor**

$$i(x) = \begin{cases} 0, & \text{if } |x| > \frac{1}{2}; \\ \frac{1}{2}, & \text{if } x = \frac{1}{2}; \\ 1, & \text{if } |x| < \frac{1}{2}. \end{cases} \tag{6}$$

  involving one sample;

- **Piecewise linear** [43]

$$i(x) = \begin{cases} 0, & \text{if } |x| > 1; \\ 1 - |x|, & \text{if } |x| \leq 1. \end{cases} \tag{7}$$

  involving two samples;

- **Four-point cubic** [44]

$$i(x) = \begin{cases} \frac{3}{2}|x|^3 - \frac{5}{2}|x|^2 + 1, & \text{if } 0 \leq |x| < 1; \\ -\frac{1}{2}|x|^3 + \frac{5}{2}|x|^2 - 4|x| + 2, & \text{if } 1 \leq |x| < 2; \\ 0, & \text{if } 2 \leq |x|. \end{cases} \tag{8}$$

  involving four samples; by this kernel, the interpolation error goes to zero uniformly at a rate proportional to the cube of the sampling increment;

- **Six-point cubic** [44]

$$i(x) = \begin{cases} \frac{4}{3}|x|^3 - \frac{7}{3}|x|^2 + 1, & \text{if } 0 \leq |x| < 1; \\ -\frac{7}{12}|x|^3 + 3|x|^2 - \frac{59}{12}|x| + \frac{15}{6}, & \text{if } 1 \leq |x| < 2; \\ \frac{1}{12}|x|^3 - \frac{2}{3}|x|^2 + \frac{21}{12}|x| - \frac{3}{2}, & \text{if } 2 \leq |x| < 3; \\ 0, & \text{if } 3 \leq |x|. \end{cases} \tag{9}$$

  involving six samples; by this kernel, the interpolation error has a fourth-order convergence rate.

- **Truncated sinc with approximate prolate spheroidal sampling window** [41]

$$i(x) = \begin{cases} \operatorname{sinc}(x) \frac{\sinh\left(\pi\nu\frac{L}{2}\sqrt{1-\left(\frac{2x}{L}\right)^2}\right)}{\sinh\left(\pi\nu\frac{L}{2}\right)\sqrt{1-\left(\frac{2x}{L}\right)^2}}, & \text{if } |x| \leq \frac{L}{2}; \\ 0, & \text{if } |x| > \frac{L}{2}. \end{cases} \tag{10}$$

where $\mathrm{sinc}(x) = \sin(\pi x)/(\pi x)$, $\nu = (1 - \frac{1}{\chi})$ and χ is the oversampling factor [18, 19]; this interpolation kernel involves an overall number of L samples; the value of L is chosen according to the desired trade-off between accuracy and computational complexity.

- **Truncated sinc with Knab sampling window** [42]

$$i(x) = \begin{cases} \mathrm{sinc}(x)\dfrac{\cosh\left(\pi\nu\frac{L}{2}\sqrt{1-\left(\frac{2x}{L}\right)^2}\right)}{\cosh\left(\pi\nu\frac{L}{2}\right)}, & \text{if } |x| \leq \frac{L}{2}; \\ 0, & \text{if } |x| > \frac{L}{2}. \end{cases} \tag{11}$$

with the same definitions of $\mathrm{sinc}(x)$ and of ν as above; as for the foregoing one, also this interpolation kernel involves an overall number of L samples, with L defining a trade-off between accuracy and complexity.

4.2. NER-NUFFT

The NUFFT algorithm employed in this paper is that developed in [9], which is based on the use of Kaiser-Bessel interpolation windows, see also [10, 11].

In order to illustrate the algorithm, let us denote by z_k the equispaced data to be transformed, by $y_l \in [-N/2, N/2]$ the N non-equispaced grid points at which calculating the transform of the z_k's and by $\hat{z}_l$ the values of the NUDFT of the z_k's. Then, the 1D NER-NUFFT algorithm quickly calculates the 1D NER-NUDFT defined as

$$\hat{z}_l = \sum_{k=-N/2}^{N/2} z_k e^{-2\pi j y_l k/N}. \tag{12}$$

The main idea of the NUFFT algorithm is to approximate the "nonuniform" exponential $\exp(-j2\pi y_l k/N)$ by interpolating few, "oversampled", "uniform" exponentials according to [9]

$$e^{-j2\pi y_l \frac{k}{N}} = \frac{(2\pi)^{-1/2}}{\Phi(2\pi k/(c_\chi N))} \sum_{m\in\mathbb{Z}} \hat{\Phi}(c_\chi y_l - m) e^{-j2\pi m \frac{k}{c_\chi N}} \tag{13}$$

where $c_\chi > 1$ is an "oversampling factor", Φ is the Kaiser-Bessel window, and $\hat{\Phi}$ is its Fourier transform.

By exploiting (13), Eq. (12) can be rewritten as

$$\hat{z}_l = \frac{1}{\sqrt{2\pi}} \sum_{m\in\mathbb{Z}} \hat{\Phi}(c_\chi y_l - m) \underbrace{\sum_{k=-N/2}^{N/2} e^{-j2\pi m \frac{k}{c_\chi N}} \frac{z_k}{\Phi(2\pi k/(c_\chi N))}}_{U_m}. \tag{14}$$

Taking now into account that $\hat{\Phi}$ has finite support, then Eq. (14) takes the form

$$\hat{z}_l \simeq \frac{1}{\sqrt{2\pi}} \sum_{p=-K}^{K} \hat{\Phi}(p) U_{p+\mu_l}, \tag{15}$$

where K is 3 or 6 for single or double precision arithmetics, respectively, μ_l is the nearest integer to $c_\chi x_l$ and the subscript $p + \mu_l$ has to be considered as $c_\chi N$-periodic.

Accordingly, the NUDFT can be effectively evaluated in three steps

(i) **Scaling and zero padding**:

$$u_k = \begin{cases} 0 & k = -c_\chi N/2, \ldots, -N/2 - 1 \\ z_k/\phi_k & k = -N/2, \ldots, N/2 - 1 \\ 0 & k = N/2, \ldots, c_\chi N/2 - 1 \end{cases}; \tag{16}$$

(ii) **FFT of** $\{u_k\}_{k=-c_\chi N/2}^{c_\chi N/2-1}$ **on** $c_\chi N$ **points to obtain** $\{U_m\}_{m=-c_\chi N/2}^{c_\chi N/2-1}$;

(iii) **Cyclic convolution to evaluate Eq. (15)**.

5. PARALLEL IMPLEMENTATION DETAILS

Under a parallel implementation, the relative values of the computational burdens of the FFT and interpolation stages can be in principle different from those involved in a sequential implementation. Analyzing this point is the aim of the present Section.

The two algorithms considered in this paper have been fully implemented in CUDA language [45] for parallel execution on a GPU and in this Section we highlight the salient points of such implementations. They share the same computational flow and instructions as in Algorithm 1, illustrated in a Matlab-like language only for the Reader's convenience. More in detail, in Algorithm 1, the

Algorithm 1 Implementation of standard and NUFFT-based BPs.

for $k = 1 : az\_bin$

$\underline{r} = \sqrt{(data.x(k) - \underline{x})^2 + (data.y(k) - \underline{y})^2 + (data.z(k) - \underline{z})^2}$

$\underline{t} = 2\underline{r}/c$

$\underline{Q}$ = **Evaluate_projection** (data.fp(:, k), $\underline{t}$); (see Eq. (4))

$\underline{A} = \underline{A} + \underline{Q}$;

end

Table 1. Processing steps of Algorithms A and B for fixed τ_n.

Step	Algorithm A	Algorithm B
#1	FFT of raw-data $s(t,\, \tau_n)$	FFT of raw-data $s(t,\, \tau_n)$
#2 (#2 & #3)	Evaluate $Q_{mn}\,(t^{U}_{mn})$ by FFT	Evaluate $Q_{mn}\,(t^{NU}_{mn})$ by NUFFT
#3	Interpolate $Q_{mn}\,(t^{U}_{mn})$ to $Q_{mn}\,(t^{NU}_{mn})$	
#4	Superimpose the result to Eq. (3)	Superimpose the result to Eq. (3)
#5	Update τ_n	Update τ_n

underscore represents array or matrices, az_bin is the overall number of sent/received pulses (see Section 2), $(data.x(k), data.y(k), data.z(k))$ is the position of the sensor for the k-th pulse, $\underline{x}$, $\underline{y}$ and $\underline{z}$ are the vectors accounting for the on-ground scene discretization, $\underline{t}$ is the vector containing the round trip traveltimes, $data.fp(:,k)$ stands for the values of $S(f,\, \tau_k)$, $\underline{Q}$ is the projection as evaluated at the k-th step and $\underline{A}$ represents the image obtained at the k-th step, and updated by the k-th projection. The "Evaluate projection" stage implements Eq. (4), corresponding to steps 2 & 3 of Table 1, and is invoked at each instance of the "for loop". The calculation of all the auxiliary quantities as well as the "Evaluate projection" stage are implemented by appropriate CUDA kernels [45]. The implementations of Algorithms A and B (see Algorithm 2) differentiate at the "Evaluate projection" stage, which is performed by an FFT + interpolation for Algorithm A and by a 1D NER-NUFFT only for Algorithm B, respectively. More in detail, the differences between the two cases reside in the *scaling_zero_padding* and *interpolation* kernels of Algorithm 2. Concerning the former, in Algorithm B, the data are scaled by a modified Bessel window before zero padding, according to [9]. Regarding the latter, the interpolation is performed by one of the kernels mentioned above in the foregoing Section for Algorithm A, while a Kaiser window is used for the NUFFT-based case, according to [9]. For both, a zero padding with a factor of c_χ (equal to 2 in our implementations) has been considered before the FFT, while the FFT step has been implemented by exploiting the CUDA cuFFT library [46, 47].

The interpolation step for the 1D NER-NUFFT has been deeply described in [11], while that for the FFT + interpolation approach has been implemented by the same CUDA kernel, exploiting different device functions corresponding to the different considered

interpolators. It should be noticed that, although nearest and linear interpolators exploit the texture memory [30, 45], this solution has not been considered here to avoid further losses of accuracy [48].

We explicitly mention that, the CUDA codes have undergone a deep optimization regarding minimizing the memory allocations [49], CPU-GPU memory transfers [50], kernel calls [51] and, most importantly, global memory accesses [35]. Finally, two versions have been implemented for each approach, one exploiting the single and one the double precision accuracy, in order to analyze the trade-off between accuracy and processing speed.

Algorithm 2 Illustrative pseudo-code for the "Evaluate_projection" stage.

```
void nufft_NER_1D_func (cufftComplex *data, float *points,
cufftComplex *result, int N, int M)
{
  /* ALLOCATIONS */
  cufftComplex *uk;
  cudaMalloc ((void **) & uk,sizeof (cufftComplex)*c_χ*N);
  /* SCALING, ZERO PADDING AND FFTSHIFT */
scaling_zero_padding ⋘ block_size 1, c_χ*N/block_size_2
+ (c_χ*N%block_size == 0 ? 0 : 1) ⋙ (data, uk, N);
  /* FFT */
  cufftHandle plan; cufftPlan1d (& plan, c_χ*N, CUFFT_C2C,
1);
  cufftExecC2C (plan, uk, uk, CUFFT_FORWARD);
  cufftDestroy (plan);
  /* INTERPOLATION */
  dim3 dimBlock(block_size_2, 1); dim3 dimGrid
(M/block_size_2 + (M%block_size_2 == 0 ? 0 : 1), 1);
  interpolation ⋘ dimGrid, dimBlock ⋙ (uk, points, result,
N, M);
  /* DEALLOCATIONS */
  cudaFree (uk);
}
```

6. NUMERICAL RESULTS

The performance of standard and NUFFT-based parallel backprojections have been assessed both in terms of accuracy and of computational speed. To this end, steps #2 and #3 of Table 1 have been run on complex random vectors of $M = 512$ elements with real and imaginary parts uniformly distributed between 0 and 1 to compute the projections at N locations, with N equal to 2^i, $i = 4, \ldots, 22$, randomly and uniformly distributed in $(-N/2, N/2)$. Reference results for steps #2 and #3 have been calculated by "direct" backprojection [4], that is, by a backprojection algorithm performing "exact" NUDFTs using Basic Linear Algebra Subroutines (BLAS) [52]. The direct backprojection has been implemented again in CUDA language by exploiting the cuBLAS library [53] (in particular, matrix-vector multiplication routines). Both the accuracy and the computational speed have been assessed following average operations on a number of 20 realizations for each individual test of interest.

As mentioned before, we distinguish between single and double precision computations. Let us stipulate that a number of interpolation samples larger than 7 are needed to achieve double precision, as in the case of the NUFFT [9]. Then

- the nearest (one sample), linear (two samples), four-point cubic (four samples), six-point cubic (six samples), Knab and approximate prolate interpolating windows with $L = 6$ samples and the NUFFT with $2K+1 = 7$ samples (i.e., the value suggested in [9] to achieve single precision) belong to the single precision class;
- Knab and approximate prolate with $L = 12$ interpolation windows and the NUFFT with $2K + 1 = 13$ interpolation samples (i.e., the value suggested in [9] to achieve double precision) belong to the double precision class.

The processing has been performed on a workstation equipped with an Intel Pentium D CPU 3.4 GHz, 4 GB of RAM, an NVIDIA Tesla C2050 [10, 11], installed on an ASUSTek P5D DH Deluxe motherboard having 2 PCI Express × 16 slots to host the GPU.

Regarding the accuracy, it can be assessed by using different criteria, as for example by resorting to coherence calculations [17–19] or to the Root Mean Square (RMS) error. Here, we compare the accuracies of the two approaches in terms of achieved RMS error between the "exact" solution and the different considered processing schemes by Figs. 2 and 3. Assessments by coherence calculations will be dealt with in Section 7. As expected, as long as the number of interpolation samples increases, a better accuracy is obtained.

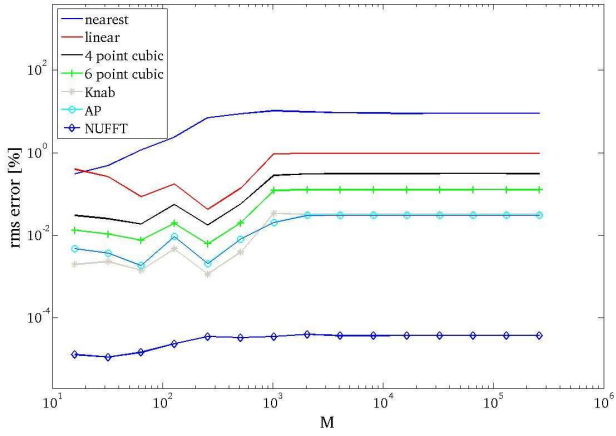

Figure 2. Rms errors of Algorithms A and B for the single precision case.

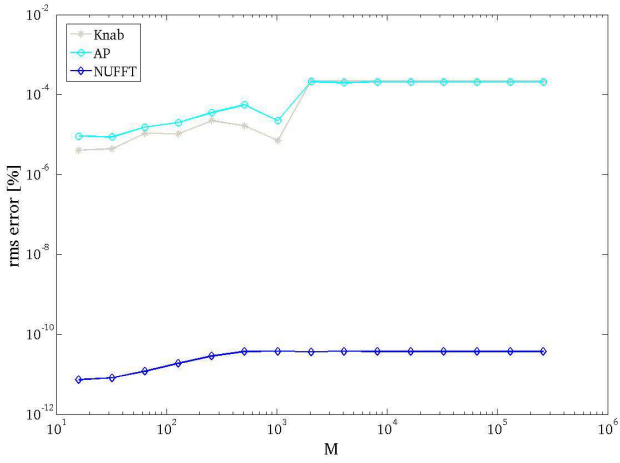

Figure 3. Rms errors of Algorithms A and B for the double precision case.

Furthermore, the NUFFT-based approach significantly outperforms the compared ones. It should be noticed that, the comparison regards the cases $N = 2^i$, $i = 4, \ldots, 18$, being the only possible ones for the cuBLAS routine to be successfully run due to memory limitations of the employed GPU.

The performance in terms of processing times[†] for the above mentioned steps #2 and #3 is illustrated in Figs. 4 and 5. As it can be

[†] The processing does not account for any CPU-GPU memory transfer.

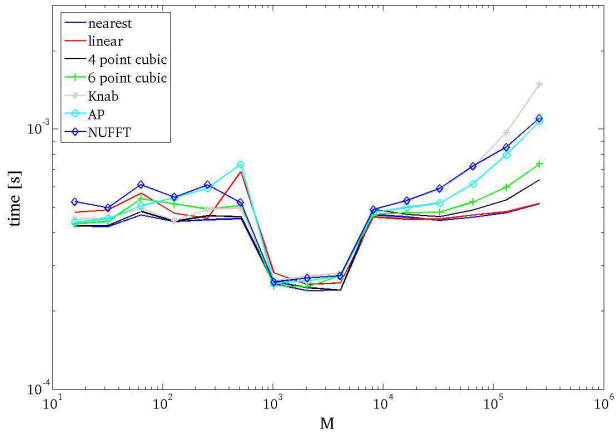

Figure 4. Processing times of Algorithms A and B for the single precision case.

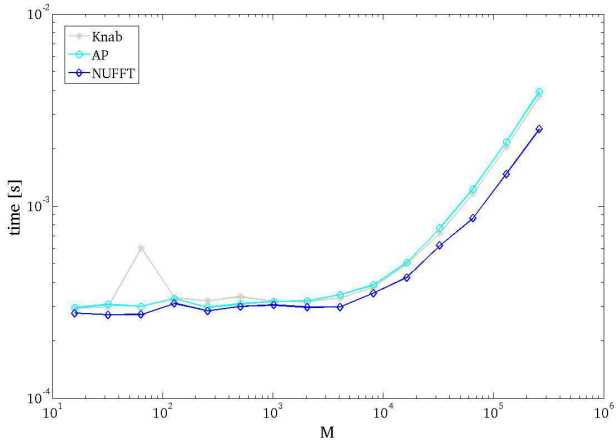

Figure 5. Processing times of Algorithms A and B for the double precision case.

seen, Algorithm B is only slightly slower than the compared ones for the single precision case, while is as fast as the compared ones for the double precision case. Again for Figs. 4 and 5, the comparison is limited to $N = 2^i$, $i = 4, \ldots, 18$. The behavior of the computation times (say, "asymptotic") for $N = 2^i$, $i = 19, \ldots, 22$ is reported in Figs. 6 and 7. Relevantly, the processing time for Algorithm B becomes less than that concerning Algorithm A exploiting the Knab or approximate prolate interpolation windows.

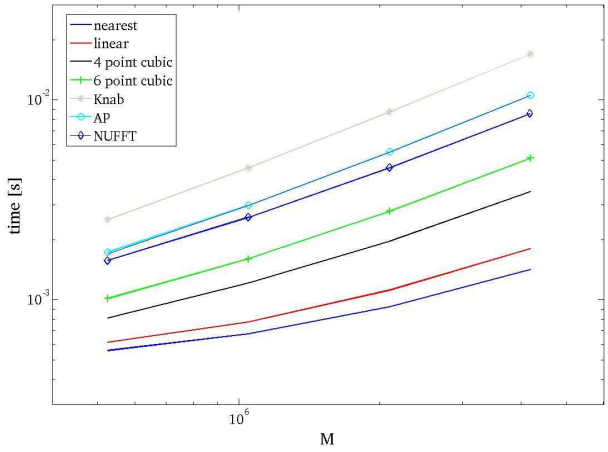

Figure 6. "Asymptotic" processing times of Algorithms A and B for the single precision case.

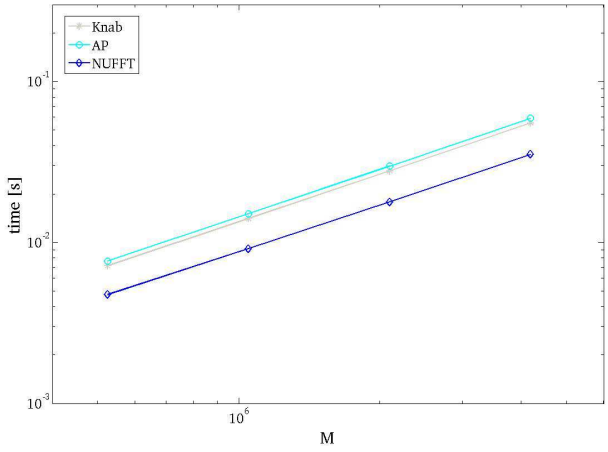

Figure 7. "Asymptotic" processing times of Algorithms A and B for the double precision case.

We finally note that, for the NUFFT-based projection evaluation and for $N = 524288$, in single precision arithmetics, approximately 16% of the time is spent in the *scaling_zero_padding* kernel, 80% of the time is due to the FFT and 4% of the time to the *interpolation* kernel, thus the performance being bounded by that of the cuFFT library. It should be also noticed that the same use of the cuFFT

library is made for Algorithms A and B, so the cuFFT processing time is the same for both the approaches. The different processing times between Algorithms A and B can be then explained by the different interpolation schemes adopted. In particular, the NUFFT-based scheme proves to be more convenient than that exploiting Knab or approximate prolate interpolation due to the use of the expansion in Eq. (13) purposely worked out for the exponentials involved in the NUDFT. Similar reasoning applies to double precision.

7. EXPERIMENTAL RESULTS

In 2009, the AFRL released X-band data collected by its own airborne-sensor in a circular acquisition geometry over an urban scene consisting of numerous buildings and civilian vehicles [38]. The purpose was to push the development of new algorithms for the detection, geolocation, tracking and identification of moving targets since, in the observed scene, multiple vehicles were driving on roads near buildings, and ground truth was provided for one of them.

The data are motion-compensated, range-compressed, delivered in the frequency domain, and cover the scenario for about 71 seconds in duration. The sensor completed two circular flight paths around the scene at HH polarization and at a height of about 7.2 km, one corresponding to a "reference" acquisition, which is used in this paper and contains no moving targets, and one corresponding to a "mission" acquisition, including the moving objects. The data have a center frequency of 9.6 GHz and a bandwidth of 640 MHz, each pulse contains 384 frequency samples and the achievable on-ground resolution is about $0.2\,\mathrm{m} \times 0.2\,\mathrm{m}$.

Figures 8 and 9 show the results obtained by using the NUFFT-based SAR backprojection algorithm (run with $K = 3$ and $K = 6$, respectively) and by considering all (i.e., 10000) the available azimuth bins, which took an overall processing time of about 45 s and 131 s, respectively. It should be noticed that we have recently implemented a multi-GPU backprojection approach which takes approximately 19 s to process the same image when run on the 2 GPUs of the front-end node of the "Jazz" cluster available at Caspur (Inter-University Consortium for the Application of Super-Computing for Universities and Research), Rome, Italy [34].

In order to compare the accuracy performance of Algorithm B to Algorithm A using the different considered interpolators, the attention has been focused on a sub-image of the result in Figs. 8 and 9. A reference sub-image has been calculated again by direct backprojection, see Fig. 10. We note that, Section 6 involves numerical data.

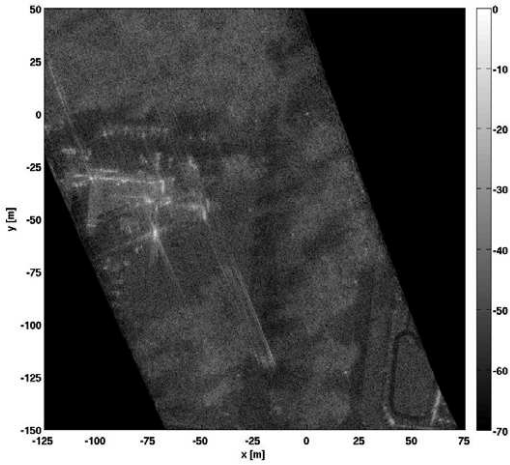

Figure 8. Algorithm B image obtained from the AFRL-GMTI data: single precision.

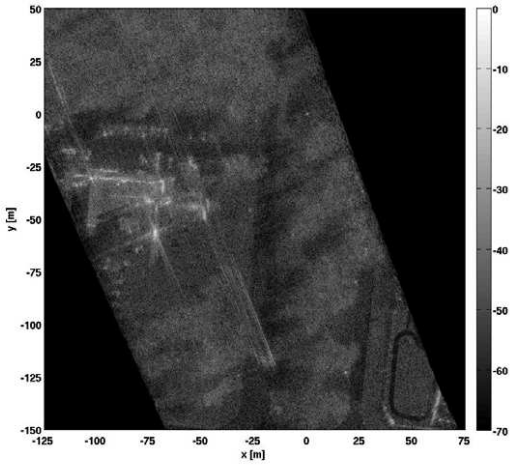

Figure 9. Algorithm B image obtained from the AFRL-GMTI data: double precision.

Therefore, evaluating the accuracy of the different implementations is performed against an "exact" reference. On the other side, Section 7 involves experimental data and an "exact" image is not available. Accordingly, the reference is obtained by the most accurate SAR

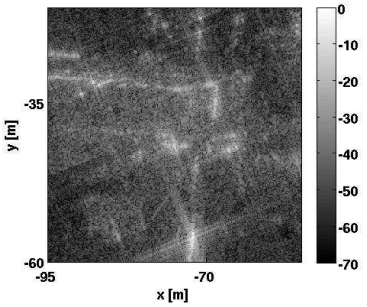

Figure 10. Reference subimage.

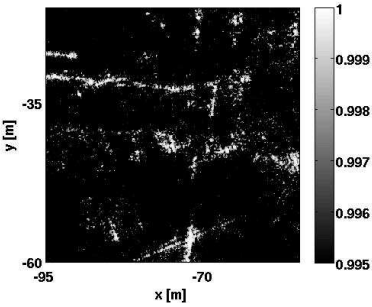

Figure 11. Single precision. Coherence maps using nearest interpolation.

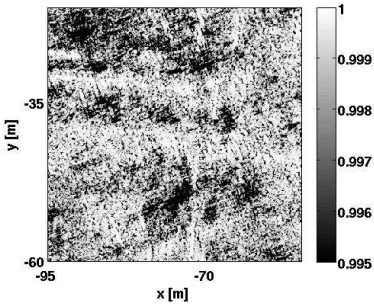

Figure 12. Single precision. Coherence maps using linear interpolation.

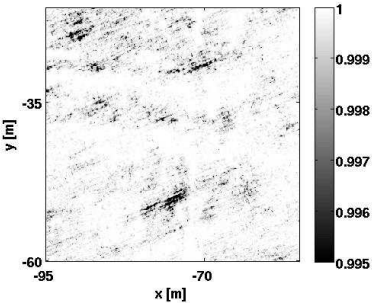

Figure 13. Single precision. Coherence maps using four point cubic interpolation.

image formation approach available. Furthermore, to assess the degree of similarity to the reference sub-image, coherence maps have been evaluated between the latter and the results achieved by the different considered schemes [22], see Figs. 11–17, for the single precision case. For the coherence maps, larger amounts of "bright" pixels are symptomatic of a larger degree of similarity to the reference sub-image and, thus, of a better accuracy. It should be noticed that the coherence maps in Figs. 11–17 have been imaged in a narrow gray scale ranging from 0.995 to 1. Indeed, a value of coherence equal to 0.995 causes an interferometric phase with a root mean square error of already 11.2°. As it can be seen from Figs. 11–17, the amount of "bright" pixels gradually increases from the nearest to the truncated sinc with approximate prolate and Knab sampling window interpolators and

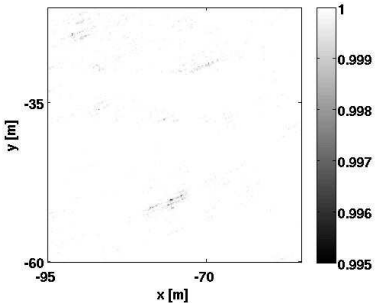

Figure 14. Single precision. Coherence maps using six point cubic interpolation.

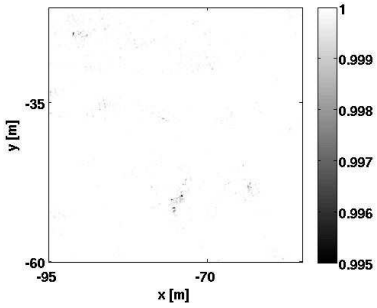

Figure 15. Single precision. Coherence maps using approximate prolate window interpolation.

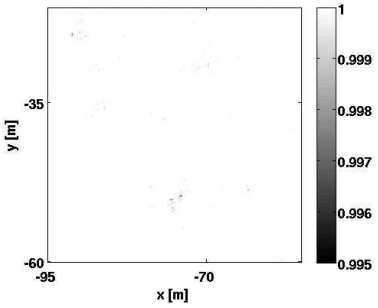

Figure 16. Single precision. Coherence maps using Knab window interpolation.

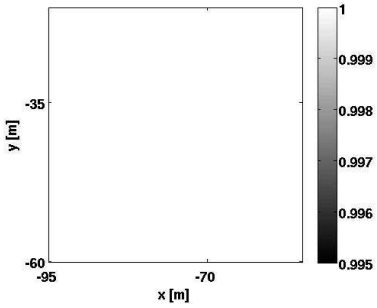

Figure 17. Single precision. Coherence maps using NUFFT.

very few dark pixels appear in the result corresponding to Algorithm B. The minimum coherence values have been equal to 0.0088, 0.5241, 0.8723, 0.9902, 0.9852, 0.9927 and 1 for the nearest, linear, 4 point cubic, 6 point cubic, approximate prolate, Knab and NUFFT cases, respectively.

Finally, due to the very high accuracy achieved by the approximate prolate and Knab window as well as the NUFFT, the minimum achieved coherence values have been, for the double precision case, equal to 1 in all the three cases. However, Algorithm B has proven faster (execution time of 23 s) than Algorithm A when using both, the approximate prolate (execution time of 31 s) or Knab (execution time of 33 s) interpolation windows. We also mention that the time taken by the GPU-based direct backprojection algorithm implemented by

double precision cuBLAS has been 650 s. Accordingly, the speedup gained by Algorithm B, equal to 28, indicates that the use of the proposed approach enables the processing of very large images, which would be otherwise unfeasible by the GPU-based direct backprojection algorithm exploiting the cuBLAS.

8. CONCLUSIONS AND FUTURE DEVELOPMENTS

We have introduced and discussed a parallel SAR backprojection algorithm profiting of the use of a 1D NER-NUFFT routine implemented on a GPU in CUDA language.

The performance of the approach has been analyzed in terms of accuracy and computational speed by comparing a version of the backprojection approach exploiting the NUFFT with a version of the same approach exploiting FFT + interpolation stages. As main results, the proposed GPU & NUFFT-based SAR backprojection:

- outperforms, in terms of accuracy, other SAR backprojection schemes based on different kinds of interpolators commonly used in SAR tomography, as nearest neighbor, piecewise linear, and four-point and six-point cubic, as well as more sophisticated and recently introduced ones, as the truncated sinc with approximate prolate spheroidal sampling window and the truncated sinc with Knab sampling window (see Figs. 2 and 3);
- shows processing times of the same order of the other confronted SAR backprojection schemes (see Figs. 4–7).

Experimental results against the AFRL challenge problem GMTI data have been also shown. The developed approach is capable to perform an accurate processing of large images in an overall computing time much more convenient than standard CPUs. Further speedups can be achieved by implementations on compact systems based on multiple GPUs [34].

It should be noticed that the computational flow in Algorithm 1 is of interest in "real-time" or "quasi real-time" processing, to enable forming SAR images as the data are acquired, pulse by pulse, as already stressed in the Introduction. Moreover, hardware like a single GPU or clusters of more than one GPU is simple, cheap and light so that it is in principle amenable of on-board installations.

The NUFFT-based backprojection approach can be extended to bistatic SAR processing [54], to fast and fast-factorized backprojection schemes [5, 7] and to the use of 2D NUFFTs (i.e., simultaneous processing of all the pulses), instead of 1D ones, when "on-line" processing is not of interest.

The approach can be finally useful for interferometric coregistration applications [18, 19, 55] and also to quickly forming 3D SAR images [56].

ACKNOWLEDGMENT

We thank the Air Force Research Laboratory for having provided the "SAR-Based GMTI in Urban Environment Challenge Problem" data sets, public release # 88 ABW-09-0967 (www.sdms.afrl.af.mil).

REFERENCES

1. Bamler, R., "A comparison of range-doppler and wavenumber domain SAR focusing algorithms," *IEEE Trans. on Geosci. Remote Sens.*, Vol. 30, No. 4, 706–713, Jul. 1992.
2. Horham, L. A. and L. J. Moore, "SAR image formation toolbox for MATLAB," *Proc. of SPIE 7699*, 769906, 2010, doi:10.1117/12.855375.
3. Desai, M. D. and W. K. Jenkins, "Convolution backprojection image reconstruction for spotlight mode synthetic aperture radar," *IEEE Trans. on Image Proc.*, Vol. 1, No. 4, 505–517, Oct. 1992.
4. Choi, H. and D. C. Munson, Jr., "Direct-Fourier reconstruction in tomography and synthetic aperture radar," *Int. J. Imaging Syst. Tech.*, Vol. 9, No. 1, 1–13, 1998.
5. Yegulalp, A. F., "Fast backprojection algorithm for synthetic aperture radar," *Proc. of the IEEE Radar Conf.*, 60–65, Waltham, MA, Apr. 20–22, 1999.
6. Basu, S. and Y. Bresler, "$O(N^2 \log_2 N)$ filtered backprojection reconstruction algorithm for tomography," *IEEE Trans. on Image Proc.*, Vol. 9, No. 10, 1760–1773, Oct. 2000.
7. Ulander, L. M. H., H. Hellsten, and G. Stenström, "Synthetic aperture radar processing using fast factorized back-projection," *IEEE Trans. on Aerosp. Electron. Syst.*, Vol. 39, No. 3, 760–776, Jul. 2003.
8. Fessler, J. A. and B. P. Sutton, "Nonuniform Fast Fourier Transforms using min-max interpolation," *IEEE Trans. on Signal Proc.*, Vol. 51, No. 2, 560–574, Feb. 2003.
9. Fourmont, K., "Non-equispaced fast Fourier transforms with applications to tomography," *J. Fourier Anal. Appl.*, Vol. 9, No. 5, 431–450, Sept. 2003.
10. Capozzoli, A., C. Curcio, A. Di Vico, and A. Liseno, "NUFFT-

& GPU-based fast imaging of vegetation," *IEICE Trans. on Commun.*, Vol. E94-B, No. 7, 2092–2103, Jul. 2011.

11. Capozzoli, A., C. Curcio, and A. Liseno, "GPU-based ω-k processing by 1D Non-Uniform FFTs," *Progress In Electromagnetic Research M*, Vol. 23, 279–298, 2012.
12. Franceschetti, G. and G. Schirinzi, "A SAR processor based on two-dimensional FFT codes," *IEEE Trans. on Aerosp. Electron. Syst.*, Vol. 26, No. 2, 356–366, Mar. 1990.
13. Cafforio, C., C. Prati, and F. Rocca, "SAR data focusing using seismic migration techniques," *IEEE Trans. on Aerosp. Electron. Syst.*, Vol. 27, No. 2, 194–207, Mar. 1991.
14. Franceschetti, G., R. Lanari, V. Pascazio, and G. Schirinzi, "WASAR: A wide-angle SAR processor," *IEE Proceedings F Radar and Signal Processing*, Vol. 139, No. 2, 107–114, Apr. 1992.
15. Franceschetti, G., A. Mazzeo, N. Mazzocca, V. Pascazio, and G. Schirinzi, "An efficient SAR parallel processor," *IEEE Trans. on Aerosp. Electron. Syst.*, Vol. 27, No. 2, 343–353, Mar. 1991.
16. Ender, J. H. G. and A. R. Brenner, "PAMIR — A wideband phased array SAR/MTI system," *IEE Proc. — Radar Sonar Navig.*, Vol. 150, No. 3, 165–172, Jun. 2003.
17. Hannsen, R. and R. Bamler, "Evaluation of interpolation kernels for SAR interferometry," *IEEE Trans. on Geosci. Remote Sens.*, Vol. 37, No. 1, 318–321, Jan. 1999.
18. Migliaccio, M. and F. Bruno, "A new interpolation kernel for SAR interferometric registration," *IEEE Trans. on Geosci. Remote Sens.*, Vol. 41, No. 5, 1105–1110, May 2003.
19. Migliaccio, M., F. Nunziata, F. Bruno, and F. Casu, "Knab sampling window for InSAR data interpolation," *IEEE Trans. on Geosci. Remote Lett.*, Vol. 4, No. 3, 397–400, Jul. 2007.
20. Capozzoli, A., G. D'Elia, A. Liseno, A. Moreira, and K. P. Papathanassiou, "A novel optimization approach to forest height reconstruction from multi-baseline data," *Proc. of the Geosci. Remote Sens. Int. Symp.*, 5037–5040, Barcelona, Spain, Jul. 23–28, 2007.
21. Capozzoli, A., G. D'Elia, A. Liseno, P. Vinetti, M. Nannini, A. Reigber, R. Scheiber, and V. Severino, "SAR tomography with optimized constellation and its application to forested scenes," *Atti della Fondazione G. Ronchi*, Vol. LXV, No. 3, 367–375, May–Jun. 2010.
22. Bamler, R. and P. Hartl, "Synthetic aperture radar interferometry," *Inverse Probl.*, Vol. 14, No. 4, R1–R54, Aug. 1998.

23. Tao, Y. B., H. Lin, and H. J. Bao, "From CPU to GPU: GPU-based electromagnetic computing (GPUECO)," *Progress In Electromagnetic Research*, Vol. 81, 1–19, 2008.
24. Dziekonski, A., A. Lamecki, and M. Mrozowski, "A memory efficient and fast sparse matrix vector product on a GPU," *Progress In Electromagnetic Research*, Vol. 116, 49–63, 2011.
25. Gao, P. C., Y. B. Tao, and H. Lin, "Fast RCS prediction using multiresolution shooting and bouncing ray method on the GPU," *Progress In Electromagnetic Research*, Vol. 107, 187–202, 2010.
26. Xu, K., Z. Fan, D.-Z. Ding, and R.-S. Chen, "GPU accelerated unconditionally stable Crank-Nicolson FDTD method for the analysis of three-dimensional microwave circuits," *Progress In Electromagnetic Research*, Vol. 102, 381–395, 2010.
27. Dziekonski, A., P. Sypek, A. Lamecki, and M. Mrozowski, "Finite element matrix generation on a GPU," *Progress In Electromagnetic Research*, Vol. 128, 249–265, 2012.
28. Demir, V., "Graphics processor unit (GPU) acceleration of finite-difference frequency-domain (FDFD) method," *Progress In Electromagnetic Research M*, Vol. 23, 29–51, 2012.
29. Di Bisceglie, M., M. Di Santo, C. Galdi, R. Lanari, and N. Ranaldo, "Synthetic aperture radar processing with GPGPU," *IEEE Signal Proc. Mag.*, Vol. 27, No. 2, 69–78, Sept. 2010.
30. Fasih, A. and T. Hartley, "GPU-accelerated synthetic aperture radar backprojection in CUDA," *Proc. of the IEEE Radar Conf.*, 1408–1413, Washington, DC, May 10–14, 2010.
31. Kusk, A. and J. Dall, "SAR focusing of P-band ice sounding data using back-projection," *Proc. of the IEEE Geosci. Remote Sens. Symp.*, 4071–4074, Honolulu, HI, Jul. 25–30, 2010.
32. Ponce, O., P. Prats, M. Rodriguez-Cassola, R. Scheiber, and A. Reigber, "Processing of circular SAR trajectories with fast factorized back-projection," *Proc. of the IEEE Geosci. Remote Sens. Int. Symp.*, 3692–3695, Vancouver, Canada, Jul. 24–29, 2011.
33. Capozzoli, A., C. Curcio, A. Liseno, and P. Vinetti, "Fast interpolation accelerated on GPU for SAR backprojection," *Proc. of the 28th Annual Rev. of Progr. in Appl. Comput. Electromagn.*, 305–310, Columbus, OH, Apr. 10–14, 2012.
34. Capozzoli, A., C. Curcio, A. Liseno, and P. V. Testa, "NUFFT-based SAR backprojection on multiple GPUs," *Proc. of the Tyrrhenian Workshop on Advances in Radar and Remote Sensing*, Napoli, Italy, Sept. 12–14, 2012.

35. Kirk, D. B. and W.-M. W. Hwu, *Programming Massively Parallel Processors*, Burlington, Morgan Kaufmann, MA, 2010.
36. Blom, M. and P. Follo, "VHF SAR image formation implemented on a GPU," *Proc. of the IEEE Int. Symp. on Geosci. Remote Sens.*, 3352–3356, Seoul, South Korea, Jul. 25–29, 2005.
37. Jackson, J. I., C. H. Meyer, D. G. Nishimura, and A. Macovski, "Selection of a convolution function for Fourier inversion using gridding," *IEEE Trans. on Med. Imag.*, Vol. 10, No. 3, 473–478, Sept. 1991.
38. Scarborough, S. M., C. H. Casteel, Jr., L. R. Gorham, M. J. Minardi, U. K. Majumder, M. G. Judge, E. Zelnio, and M. Bryant, "A challenge problem for SAR-based GMTI in urban environments," *Proc. of SPIE 7337*, 73370G, 2009, doi:10.1117/12.823461.
39. Hein, A., *Processing of SAR Data: Fundamentals, Signal Processing, Interferometry*, Springer-Verlag, Berlin, Heidelberg, 2004.
40. D'Elia, G., G. Leone, R. Pierri, and G. Schirinzi, "Traveling sampling of scattered fields," *Proc. of the IEEE Antennas Propag. Int. Symp.*, 531–534, Vancouver, Canada, Jun. 17–21, 1985.
41. Knab, J. J., "Interpolation of band-limited functions using the approximate prolate series," *IEEE Trans. on Inf. Theory*, Vol. 25, No. 6, 717–719, Nov. 1979.
42. Knab, J. J., "The sampling window," *IEEE Trans. on Inf. Theory*, Vol. IT-29, No. 1, 157–159, Jan. 1983.
43. Li, A., "Algorithms for the implementation of Stolt interpolation is SAR processing," *Proc. of the IEEE Geosci. Remote Sens. Symp.*, 360–362, Houston, TX, May 26–29, 1992.
44. Keys, R. G., "Cubic convolution interpolation for digital image processing," *IEEE Trans. on Acoust. Speech Signal Proc.*, Vol. 29, No. 6, 1153–1160, Dec. 1981.
45. Sanders, J. and E. Kandrot, *CUDA by Example*, Addison-Wesley, Ann Arbor, MI, 2011.
46. CUDA CUFFT Library, Feb. 2011.
47. Nukada, A. and S. Matsuoka, "Auto-tuning 3-D FFT library for CUDA GPUs," *Proc. of Conf. on High Performance Computing Networking, Storage and Anal.*, Portland, OR, Nov. 14–20, 2009.
48. Ruijters, D., B. M. ter Haar Romeny, and P. Suetens, "Efficient GPU-based texture interpolation using uniform B-splines," *J. Graphics, GPU, Game Tools*, Vol. 13, No. 4, 61–69, Jan. 2008.
49. http://www.cs.virginia.edu/∼ mwb7w/cuda_support/memory_m-

anagement_overhead.html.

50. http://www.cs.virginia.edu/~mwb7w/cuda_support/memory_transfer_overhead.html.
51. Tarjan, D., K. Skadron, and P. Micikevicius, "The art of performance tuning for CUDA and manycore architectures," Birds-of-a-feather session at SC'09, 2009.
52. Capozzoli, A., C. Curcio, G. D'Elia, A. Liseno, and P. Vinetti, "Fast CPU/GPU pattern evaluation of irregular arrays," *Appl. Comput. Electromagn. Soc. J.*, Vol. 25, No. 4, 355–372, Apr. 2010.
53. CUDA Toolkit 4.0, CUBLAS Library, Apr. 2011.
54. Rigling, B. D. and R. L. Moses, "Polar format algorithm for bistatic SAR," *IEEE Trans. on Aerosp. Electron. Syst.*, Vol. 40, No. 4, 1147–1159, Oct. 2004.
55. Selva, J. and J. M. Lopez-Sanchez, "Efficient interpolation of SAR images for coregistration in SAR interferometry," *IEEE Geosci. Remote Sens. Lett.*, Vol. 4, No. 3, 411–415, Jul. 2007.
56. Austin, C. D., E. Ertin, and R. L. Moses, "Sparse multipass 3D SAR imaging: Applications to the GOTCHA data set," *Proc. of SPIE Algorithms for Synthetic Aperture Radar Imagery XVI*, Vol. 7337, Orlando, FL, Apr. 16–17, 2009.