A Fast Decimation-in-image Back-projection Algorithm for SAR

Shaun I. Kelly and Mike E. Davies
Institute for Digital Communications
The University of Edinburgh
email: {Shaun.Kelly, Mike.Davies}@ed.ac.uk

***Abstract*— Fast back-projection algorithms are required for new modalities of SAR, such as UWB SAR. In this paper we propose a novel algorithm which we call the fast decimation-in-image back-projection algorithm due to its relation to decimation-in-time FFT algorithms. It is the natural dual of existing fast back-projection algorithms which are related to decimation-in-frequency FFT algorithms. The proposed algorithm provides similar speed up to existing algorithms, however, it has additional advantages. The advantages relate to the way in which the algorithm manifests errors. The size and nature of the errors introduced in the proposed algorithm are more desirable than that of existing algorithms.**

I. Introduction

The Synthetic Aperture Radar (SAR) imaging methodology was developed as a more effective means of producing radar images than *real-aperture* imaging methods. Unlike a real-aperture imaging system, a SAR system does not require a large cross-range antenna aperture in order to produce a high cross-range resolution image. It instead uses a smaller physical aperture antenna mounted to a moving *platform* (typically an aircraft or a satellite) to synthesise a larger aperture.

The goal of SAR imaging is to form an image, which will be an approximation of the scene reflectivities. This image will consist of a finite number of complex samples. The majority of SAR image formation algorithms, can be consider to be an approximation of a two-dimensional match filter, where each pixel in the resulting image is the result of a correlation of the phase history with the expected signal for a target at that location.

For airborne SAR, arguably the most commonly used image formation algorithms are the back-projection (BP) algorithm, the range migration algorithm (RMA) and the polar format algorithm (PFA). Each algorithm has its advantages and disadvantages, which are based around the trade off between image quality and computational complexity. The most accurate approximation to the true match filter solution is achieved by the BP algorithm. The BP algorithm can be used for all types of synthetic apertures, chirp bandwidths, beamwidths and scene terrains. In contrast, the RMA is only suitable for flat terrain and synthetic apertures that are linear. While, the PFA is only suitable for far-field imaging. Unfortunately, the complexity of the BP algorithm is much greater than the complexity of the other algorithms and therefore it has seldom been used operationally.

New modalities of SAR, in particular ultra-wideband (UWB) SAR, have renewed interest in the BP algorithm. In these applications the synthetic aperture can be very long which makes maintaining a linear aperture challenging. Also, at large wavelengths, the antenna beam width is usually large, making the imaging near-field. In these scenarios, the BP algorithm is the only match filter based algorithm suitable for image formation.

These types of applications have led many researchers to develop algorithms, which are as versatile as the BP algorithm, but have a reduced complexity [1], [2], [3]. Fast BP algorithms have been proposed which require $\mathcal{O}\left(N^2 \log N\right)$ operations for an $N \times N$ pixel image which is formed from a phase history with N aperture positions. This is a reduce order of operations, when compared to the BP algorithm which requires $\mathcal{O}\left(N^3\right)$ operations. The reduce complexity order is usually achieved by recursively splitting the image grid and correspondingly decimating the phase history before performing the BP algorithm. In [1], the image grid is recursively split in a quad tree structure and the phase history is decimated by combining close aperture positions at each stage of recursion, with a beamforming like procedure in a polar co-ordinate system. In [2], [3], the image is also recursively split in a quad tree structure, however, the phase history is decimated at each stage of recursion in a different manner to the previous algorithms. The phase history is instead decimated independently in fast-time and slow-time. The algorithm presented in [3] was motivated by [4], which appeared in the tomography literature, but is virtually identical to the algorithm presented in [2]. More recently, in [5], another algorithm has been proposed which has a different strategy for achieving the same order of complexity. The authors state that their "butterfly" algorithm is a generalisation of the fast multipole method (FMM) [6]. The FMM has previously been used to compute fast matrix vector products, for example the FMM based non-uniform FFT (NFFT) [7].

In this paper, a novel fast BP algorithm will be proposed, which is coined, the fast *decimation-in-image* BP algorithm. The name was selected due to its relation to decimation-in-time FFT algorithms. It is the natural dual of the other fast BP algorithms [2], [3] which we coin as fast *decimation-in-phase-history* BP algorithms, due to their relation to decimation-in-frequency FFT algorithms. The performance of the fast decimation-in-image BP algorithm in terms of its image qual-

978-1-4799-2035-8/14/$31.00@2014 IEEE

ity and computational speed will demonstrated.

II. Image Formation

Standard SAR image formation algorithms are based on approximations of a two-dimensional match filter. To compute this match filter we must first define the forward observational model of the SAR system. An observational model for a monostatic SAR system after it has been dechirped and deskewed is given by

$$\begin{aligned}\mathbf{Y} &= h\left(\mathbf{X}\right) \\ &= \left\{ \sum_{k=1}^{K} \sum_{l=1}^{L} b_\theta\left(\vec{\mathbf{y}}_{kl}, \vec{\mathbf{x}}_n\right) b_\phi\left(\vec{\mathbf{y}}_{kl}, \vec{\mathbf{x}}_n\right) a_{kln} x_{kl} \right. \\ &\qquad \left. \exp\left(-j u_{kln} k_m\right) \right\}_{mn},\end{aligned} \tag{1}$$

where,

- $\mathbf{Y} \in \mathbb{C}^{M \times N}$ is the fast-time by slow-time phase history,
- $\mathbf{X} \in \mathbb{C}^{K \times L}$ are the reflectivities of scene elements,
- $\vec{\mathbf{x}}_n = \left[x_n^1, x_n^2, x_n^3\right]^{\mathrm{T}}$ is the position of the platform at the n^{th} aperture location,
- $\vec{\mathbf{y}}_{kl} = \left[y_{kl}^1, y_{kl}^2, y_{kl}^3\right]^{\mathrm{T}}$ is the position of each of the scene elements,
- $b_\theta\left(\vec{\mathbf{y}}_{kl}, \vec{\mathbf{x}}_n\right)$ models the antenna beam pattern in the polar angle direction,
- $b_\phi\left(\vec{\mathbf{y}}_{kl}, \vec{\mathbf{x}}_n\right)$ models the antenna beam pattern in the azimuth angle direction,
- $u_{kln} = \left\|\vec{\mathbf{y}}_{kl} - \vec{\mathbf{x}}_n\right\| - u_0\left(\vec{\mathbf{x}}_n\right)$ is the distance between the platform and an element in the scene, relative to the dechirp point $u_0\left(\vec{\mathbf{x}}_n\right)$,
- $a_{kln} = \left(4\pi\left(u_0\left(\vec{\mathbf{x}}_n\right) + u_{kln}\right)\right)^{-2}$ models the RF wave energy loss due to spreading in three-dimensional space,
- $k_m = 2\left(\omega_0 + 2\alpha\left(\left(m-1\right)T_{\mathrm{s}} - \frac{T}{2}\right)\right)/c_0$,
- c_0 is the speed of light in vacuum,
- ω_0 is the carrier frequency,
- 2α is the chirp rate,
- T_{s} is fast-time sample rate and
- T is the chirp period.

The match filter for this forward observational model $h\left(\cdot\right)$ is given by its adjoint $h^{\mathrm{H}}\left(\cdot\right)$. The direct computation of $h^{\mathrm{H}}\left(\cdot\right)$ can be written as

$$\begin{aligned}\hat{\mathbf{X}} = \left\{ \sum_{n=1}^{N} b_\theta\left(\vec{\mathbf{y}}_{kl}, \vec{\mathbf{x}}_n\right) b_\phi\left(\vec{\mathbf{y}}_{kl}, \vec{\mathbf{x}}_n\right) a_{kln} \right. \\ \left. \sum_{m=1}^{M} y_{mn} \exp\left(j k_m u_{kln}\right) \right\}_{kl}.\end{aligned} \tag{2}$$

The formed image, $\hat{\mathbf{X}}$, can be computed in this direct manner in $\mathcal{O}\left(KLMN\right)$ operations. This order of operations is much too high for any realistically sized problems. Therefore, reduced complexity algorithms are used to approximate the adjoint.

A. Back-projection Algorithm

The BP algorithm reduces the complexity of computing the adjoint by first observing that the summation in Eq. 2 can be written as

$$\hat{\mathbf{X}} = \left\{ \sum_{n=1}^{N} b_\theta\left(\vec{\mathbf{y}}_{kl}, \vec{\mathbf{x}}_n\right) b_\phi\left(\vec{\mathbf{y}}_{kl}, \vec{\mathbf{x}}_n\right) a_{kln} p_{kln} \right\}_{kl}, \tag{3}$$

where,

$$p_{kln} = \sum_{m=1}^{M} y_{mn} \exp\left(j k_m u_{kln}\right). \tag{4}$$

The reduce complexity is then achieved by approximating the elements $\left\{p_{kln}\right\}_{kl}$. This approximation is possible because the summation in Eq. 4 is a non-uniform DFT. Therefore, the elements of the matrix $\mathbf{P}_n = \left\{p_{kln}\right\}_{kl}$ can be approximated using a NFFT algorithm. Traditionally, zero padding and linear interpolation has been used to approximate $\mathbf{P}_n$ in $\mathcal{O}\left(M\log\left(M\right) + KL\right)$ operations. Modern NFFT algorithms can also be used to approximate $\mathbf{P}_n$ [8]. They have a complexity order which dependents on the approximation error. To compute $\mathbf{P}_n$ with these algorithms requires $\mathcal{O}\left(M\log\left(M\right) + KL\log\left(1/\epsilon\right)\right)$ operations, where ϵ is the approximation error. Using this approximation for $\mathbf{P}_n$, $\hat{\mathbf{X}}$ can be approximated in $\mathcal{O}\left(KLN\log\left(1/\epsilon\right)\right)$ operations.

B. Fast Back-projection Algorithms

The fast BP algorithms in [2], [3] recursively decimate the phase history and split the image grid at each stage of recursion. At the final stage of recursion, the BP algorithm is performed on a number of decimated phase histories to produce each segment of the final image. This is visualised in Fig. 1(a). This can be seen to be analogous with the decimation-in-frequency FFT algorithms. In this section we will show how an equivalent fast decimation-in-image BP algorithm can be used which is analogous with the decimation-in-time FFT algorithms.

1) Fast Decimation-in-image Back-Projection Algorithm: The decomposition strategy used in the fast decimation-in-image BP algorithm using a quad-tree structure is visualised in Fig. 1(b). At the top of the figure we have the original phase history and its corresponding image grid. At the first stage of decomposition, seen on the second line of the figure, the phase history is split into four even segments and the image is decimated into four new phase histories, with each corresponding to one of the four image segments. At the second stage, this decomposition is repeated so there are now 16 phase history grid segments and there are 16 corresponding decimated images. At the final stage of decomposition, the BP algorithm is used to compute an image for each segment of the phase history. The images can then be recursively upsampled and combined to produce a single image.

Decimating in the image domain is possible because the sampling rate of the scene is dependent on the period of the slow and fast times. In order to provide an analytic

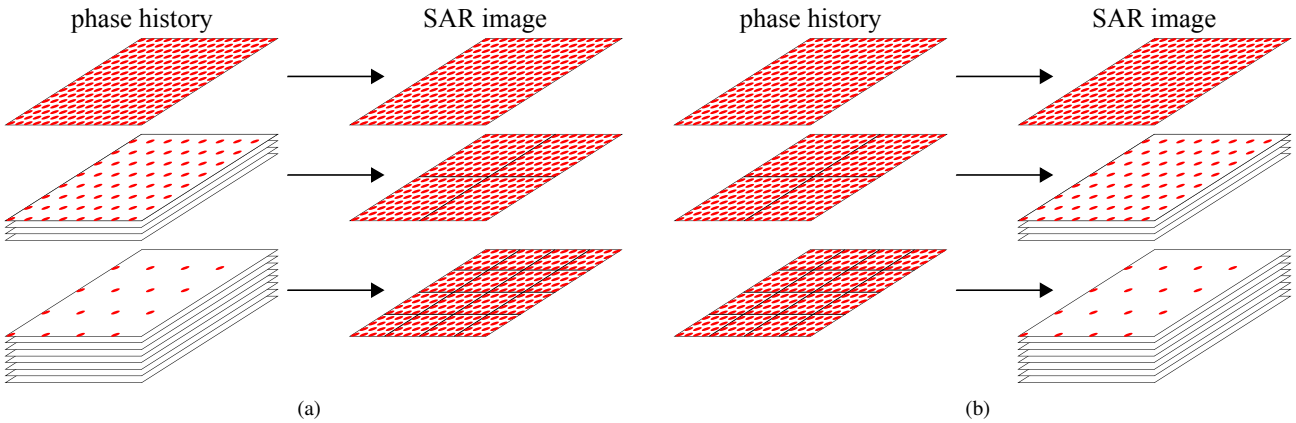


Fig. 1. Decomposition in the fast BP Algorithms. (a) Decomposition in the decimation-in-phase-history algorithm. (b) Decomposition in the decimation-in-image algorithm.

expression, we will make some approximations to simplify this relationship. It is important to note that these approximation are only used to simplify the analysis of the algorithm and are not required in order to use the algorithm in practice. We will first assume that the scene elements lie on the plane $\vec{\mathbf{y}} = \left[y^1, y^2, 0\right]^{\mathrm{T}}$. We will also make a far-field approximation and assume that aperture is in the y^2 direction. Finally we will assume that the slow and fast times are small enough such that wavenumber spectral support of the image is approximately rectangular. Under these approximations we can provide a bounds for the minimum sampling rates for the image grid, which are given by,

$$f_{y^1\mathrm{min}} \leq \frac{2\alpha}{\pi c_0} \sin\left(\theta_0\right) T \tag{5a}$$

$$f_{y^2\mathrm{min}} \leq \frac{\|\vec{\mathbf{v}}\| \omega_0}{\pi c_0 u_0} \sin\left(\theta_0\right) T_{\hat{t}}, \tag{5b}$$

where, $\|\vec{\mathbf{v}}\|$ is the platform velocity, $T_{\hat{t}}$ is slow-time period and θ_0 is the polar angle of the antenna beam centre. In this scenario both sampling rates are linearly dependent on the SAR image radius. For the more general case where these approximations do not apply we do not have a linear dependency, however, it is still possible to have a practical decomposition strategy. This is demonstrated in Fig. 2 using a simulated phase history. In this simulation, the phase history has been split into four segments. The BP algorithm was used to generate four images, one from each of the phase histories. An image was also created from the full phase history. For the complete phase history, the required sampling rates are approximately 1.75 samples per metre in the y^1 direction and 2.2 samples per metre in the y^2 direction. For the phase history segments in Fig. 2(c) and Fig. 2(e), the required sampling rates are approximately 0.9 and 0.7 samples per metre in each of the directions. For the phase history segments in Fig. 2(d) and Fig. 2(f), the required sampling rates are approximately 1 and 1.1 samples per metre in each of the directions. This simulation shows that the required sampling rates for the image grid, approximately increases linearly with the period of the slow and fast times.

This property is used in the fast decimation-in-image BP algorithm. When the phase history is split into four image segments, the required sampling rates of the image are approximately halved. Therefore the image grid can be decimated. If the phase history is recursively split, such that each phase history segment contains just a single element, the number of operations required to perform the BP algorithm for all the images will be $\mathcal{O}\left(MN\right)$. The next stage of the fast decimation-in-image BP algorithm is to combine the images. This combination can be done recursively. At each stage, groups of four images, which were formed from adjacent phase history segments, are combined. To do this each image must first be upsampled by a factor of two in each direction. This can be done, for all decimated images, with $\mathcal{O}\left(KL\right)$ operations. The images can then be combined. This procedure is repeated until there is just a single image. The number of operations required to combine all images into a single image will be $\mathcal{O}\left(MN \log\left(\max\left\{M, N\right\}\right)\right)$ operations. Therefore the complexity of the fast decimation-in-image BP algorithm is $\mathcal{O}\left(MN \log\left(\max\left\{M, N\right\}\right)\right)$.

A detailed description of the proposed algorithm, with a single stage of decomposition, is given in Algorithm 1, where, $\mathbf{h} \in \mathbb{C}^{M_{\mathrm{h}} \times N_{\mathrm{h}}}$ is a two-dimensional low pass filter and

$$\mathrm{nfft}_n\left(\mathbf{Y}'\right) \approx \sum_{m=1}^{M} y'_{mn} \exp\left(jk_{\mathrm{u}}\left(t_m\right) u_{kln}\right) \tag{6}$$

is an approximation of the nonuniform DFT.

There are errors which are introduced by approximating the adjoint with a fast BP algorithm. In both algorithms, the main source of error is due to the decimation or upsampling of finite length data with finite length filters. Since the data (the phase history or the image) is finite length, any decimating filter or upsampling filter will have "border effects" which will produce errors. Also, since in practise a filter will be finite length, the filter will have associated stopband and passband ripples and a transition bandwidth, which will introduce errors.

Algorithm 1 fast decimation-in-image BP algorithm $\hat{\mathbf{X}} = g_{\text{fbp}}(\mathbf{Y})$

Input: $\mathbf{Y}$
Output: $\hat{\mathbf{X}}$

$\mathbf{Y} \leftarrow \mathbf{Y}$
$\hat{\mathbf{X}} \leftarrow \mathbf{0}_{K,L}$
for $a = 0, 1$ **do**
  for $b = 0, 1$ **do**
    {**SPLIT PHASE-HISTORY**}
    $\mathbf{Y}' \leftarrow \left\{ y_{(aM/2+m)(bN/2+n)} \right\}_{m=1,n=1}^{M/2,N/2}$
    $\vec{\mathbf{x}}' \leftarrow \left\{ \vec{\mathbf{x}}_{bN/2+n} \right\}_{n=1}^{N/2}$
    {**BACK-PROJECTION ALGORITHM**}
    for $n = 1, \cdots, N/2$ **do**
      $\hat{\mathbf{X}}' \leftarrow \mathbf{0}_{K/2,L/2}$
      $\mathbf{P} \leftarrow \text{nfft}_n\left(\mathbf{Y}'\right)$
      for $k = 1, \cdots K/2$ **do**
        for $l = 1, \cdots, L/2$ **do**
          $\vec{\mathbf{y}}' \leftarrow \vec{\mathbf{y}}_{(2k-1)(2l-1)}$
          $\hat{x}'_{kl} \leftarrow \hat{x}_{kl} + a_{(2k-1)(2l-1)n} p_{kl} \times$
            $b_\theta\left(\vec{\mathbf{y}}_{(2k-1)(2l-1)}, \vec{\mathbf{x}}_n\right) \times$
            $b_\phi\left(\vec{\mathbf{y}}_{(2k-1)(2l-1)}, \vec{\mathbf{x}}_n\right)$
        end for
      end for
    end for
    {**CENTRE IMAGE'S SPECTRAL SUPPORT**}
    $\vec{\mathbf{x}}_0 \leftarrow \vec{\mathbf{x}}_{\lceil N/2 \rceil + 1}$
    $t_0 \leftarrow t_{\lceil M/2 \rceil + 1}$
    $\hat{\mathbf{X}}' \leftarrow \left\{ \hat{x}'_{kl} \exp\Big(-jk_{\text{u}}\left(t_0\right) \times \right.$
      $\left(\left\| \vec{\mathbf{y}}_{(2k-1)(2l-1)} - \vec{\mathbf{x}}_0 \right\| - \right.$
      $\left. \left. \| \vec{\mathbf{y}}_0 - \vec{\mathbf{x}}_n \| \right) \right) \Big\}_{k=1,n=l}^{K/2,L/2}$
    {**UPSAMPLING**}
    $\hat{\mathbf{X}}' \leftarrow \{$if $k \wedge l$ are odd then $x'_{(k/2+1/2)(l/2+1/2)}$,
      else $0\}_{k=1,n=l}^{K,L}$
    $\hat{\mathbf{X}}' \leftarrow \left\{ \sum_{k'=1}^{2M_h+1} \sum_{l'=1}^{2N_h+1} \right.$
      $\left. h_{k'l'} \hat{x}'_{(2k-1-M_h+k')(2l-1-N_h+l')} \right\}_{k=1,l=1}^{K,L}$
    {**COMBINE IMAGES**}
    $\hat{\mathbf{X}}' \leftarrow \left\{ \hat{x}'_{kl} \exp\big(jk_{\text{u}}\left(t_0\right) \right.$
      $\left. \left(\| \vec{\mathbf{y}}_{kl} - \vec{\mathbf{x}}_0 \| - \| \vec{\mathbf{y}}_0 - \vec{\mathbf{x}}_n \| \right) \right) \Big\}_{k=1,n=l}^{K,L}$
    $\hat{\mathbf{X}} = \hat{\mathbf{X}} + \hat{\mathbf{X}}'$
  end for
end for

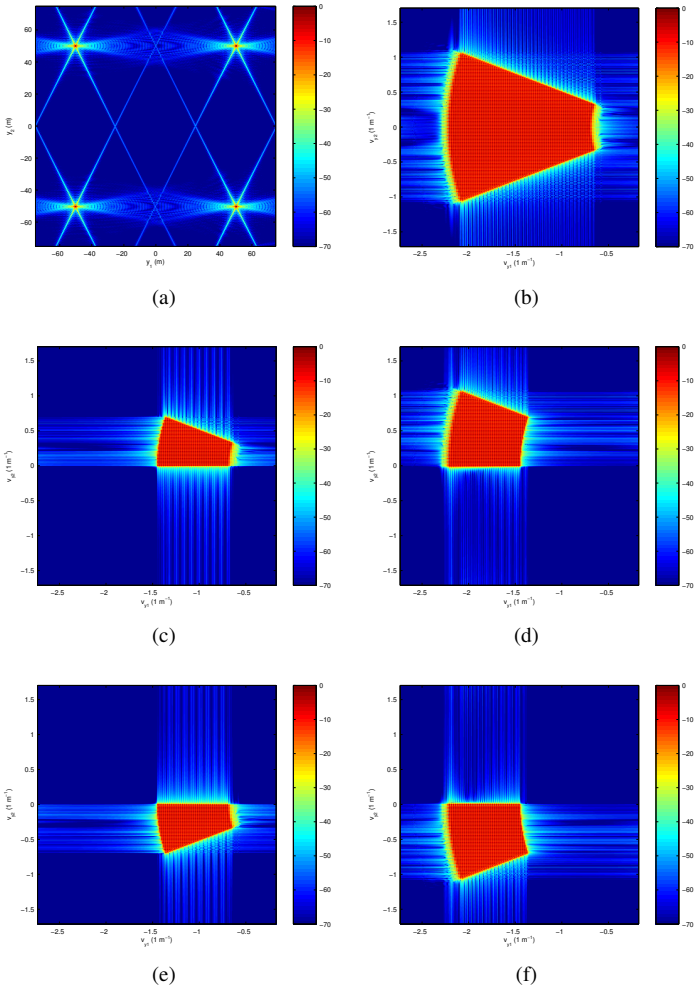


Fig. 2. Images formed using the BP algorithm from a simulated phase history which was generated using four targets located at $[25, 25, 0]^T$ m, $[-25, 25, 0]^T$ m, $[25, -25, 0]^T$ m and $[-25, -25, 0]^T$ m (relative to the scene centre). The system parameter are: carrier frequency $\omega_0/2\pi = 308$ MHz, chirp rate $\alpha/\pi = 32.4$ MHz/μs, chirp period $T = 10$ μs, synthetic aperture $[7, -3.5, 7]^T$ km to $[7, 3.5, 7]^T$ km (relative to the scene centre) and platform velocity 100 m/s. (a) is the resulting image and (b) is its spectrum. (c), (d), (e) and (f) are the spectra of images formed using four evenly sized segments of the full phase history.

In the decimation-in-phase-history algorithm, since the decimation is performed on the phase history, the border effects and finite filter length errors will occur in the phase history domain. For the decimation-in-image algorithm, because upsampling is performed on the image, the border effects and finite filter length errors will occur in the image domain.

III. Experiments

To demonstrate the numerical performance of the proposed fast decimation-in-image BP algorithm, 4° of the GOTCHA data set was used to form three images [9]. One using the BP algorithm and two others using the two fast BP algorithms with three stages of decomposition. The BP algorithm used the NFFT algorithm in [10], with an interpolation kernel length of 24 samples which is what was suggested for double

TABLE I

IMAGE FORMATION TIMES (SECONDS)

N	BP	fast BP (dec.-in-image)	fast BP (dec.-in-phase-history)	PFA
256	18.26	4.75	4.64	0.90
512	140.22	18.77	18.74	3.24
1024	1120.96	77.18	76.98	13.15
2048	9052.47	318.43	317.36	69.15

precision numerical accuracy. The filter used for decimating and upsampling, in both of the fast algorithms, was a 41 sample length low-pass Chebyshev filter with 100dB side-lobe attenuation.

The BP image is shown in Fig. 3. Visually, there is very little difference between the three images. Therefore, we show the relative errors between the fast BP algorithm images and the BP algorithm image in Fig. 4. Due to the source of the errors, the relative errors are displayed in the wavenumber domain for the decimation-in-phase-history image and in the image domain for the decimation-in-image image. The relative error $\|\hat{x} - x\|_2 / \|x\|_2$ is computed for each element in the images, where, x is an element from the reference BP image and $\hat{x}$ is the corresponding element from fast BP algorithm image.

The results in Fig. 4(a), demonstrate the errors in the images formed using the fast decimation-in-image BP algorithm. With three stages of decomposition, the errors within the centre of the image are approximately -90 dB. There are also observable border effects.

For the fast decimation-in-phase-history BP algorithm, the results in Fig. 4(b) demonstrate the errors in the formed images. With three stages of decomposition, the errors within the spectral support of the image are approximately -40 dB and the edge effects are also observable. The errors associated with border effects, unlike in the decimation-in-image images, are not concentrated on the edges of the spectral support of the image. This is because targets have a spatially varying spectral support in an image formed using the BP algorithm.

To demonstrate the computational advantages of the fast BP algorithms, the computational times for image formation were measured for different size phase histories and images. Images were formed using the BP algorithm, the PFA and the fast decimation-in-phase-history and decimation-in-image BP algorithms. Both the BP algorithm and the PFA made use a NFFT algorithm with an interpolation kernel length of 24 samples. The image formation times were measured on a single core of a 2.5 GHz Intel Xeon processor with N^2 element images and N^2 element phase histories. The number of decomposition stages in the fast BP algorithms was $\log_2 N - \log_2 64$. This number of stages was selected because through numerical simulation it was found to be a good trade off between algorithm "speed up" and approximation error. Table I shows the resulting image formation times in seconds.

For small images and phase histories the fast BP algorithms only provide modest speed up when compared to the BP algorithm. However, as the problem size grows, the speed up becomes more significant. It is also interesting to note that the ratio of the image formation times for the PFA and the fast BP algorithms is approximately constant for all problem sizes. Since both algorithms have the same theoretical order of operations, this is an expected result.

It is also worth noting that the BP algorithm and the PFA were implemented in `C` code while the decimation and upsampling code of the fast BP algorithms were implemented in `Matlab` script. If this code was ported to `C` code, one would expect some additional speed up.

IV. Conclusion

In this paper we have proposed a novel fast BP algorithm. Fast BP algorithms are important for new modalities of SAR, such as UWB SAR. The fast decimation-in-image BP algorithm provides a similar level of speed up to existing algorithms, however, it has other advantages. Firstly, the errors introduced by the algorithm are concentrated on the edges of the image. This region usually contains the smallest amount of energy in the image due to the antenna beam pattern. Therefore, making the errors less significant. Also, since decimation and upsampling is most easily performed on a uniform grid, if the aperture is irregularly sampled, decimation-in-phase-history algorithms must first interpolate onto a uniform grid before decimation. This is not required in the decimation-in-image algorithm because the image grid is defined to be uniform. This could be a significant advantage for the decimation-in-image algorithm. This is because, like the standard BP algorithm, the algorithm naturally accounts for non-ideal platform motion.

Like other fast BP algorithms, an equivalent fast observational model (a fast re-projection algorithm) could be produce and the two algorithms could be used in an iterative image formation algorithm [11]. Also Graphical Processing Units (GPU) could be used for additional speed up since the algorithms are inherently parallelisable.

Further work could investigate decimation and upsampling strategies that reduce the border effects. This would include investigating the design of improved finite length filters and possibly non-uniform sample grids. For example, the butterfly algorithm in [5] uses Chebyshev nodes.

V. Acknowledgements

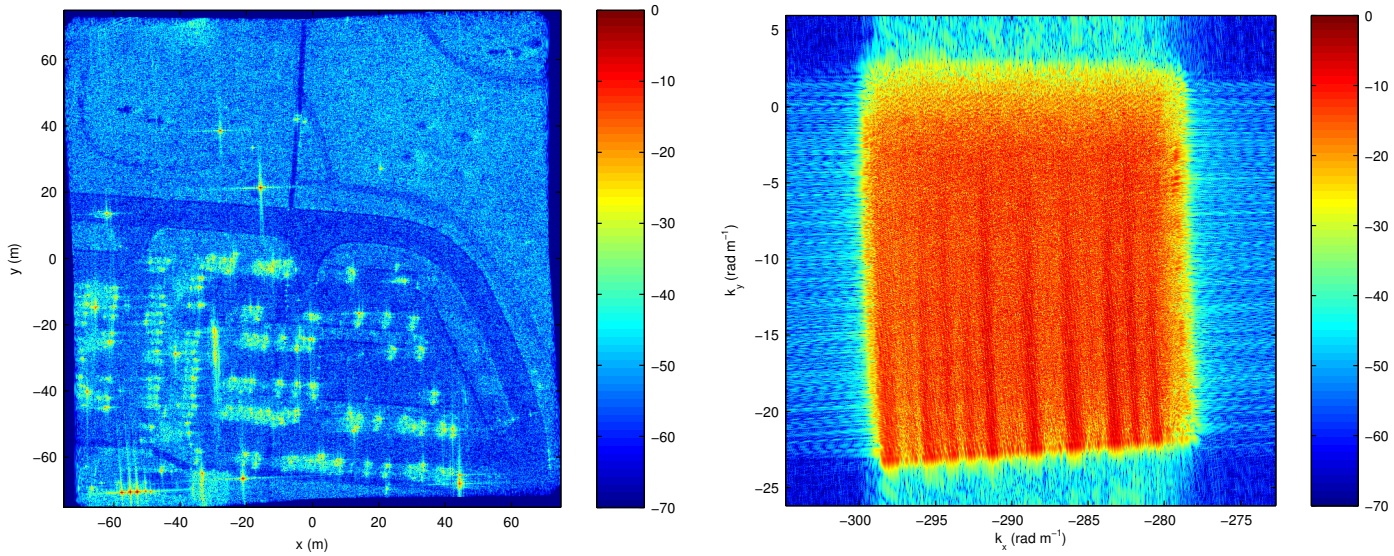
This work forms part of the UK University Defence Research Collaboration in Signal Processing and was supported by EPSRC/DSTL grants [EP/H012370/1, EP/J015180/1 and EP/K014277/1].

References

[1] L. Ulander, H. Hellsten, and G. Stenstrom, "Synthetic-aperture radar processing using fast factorized back-projection," vol. 39, no. 3, pp. 760–776, Jul. 2003.

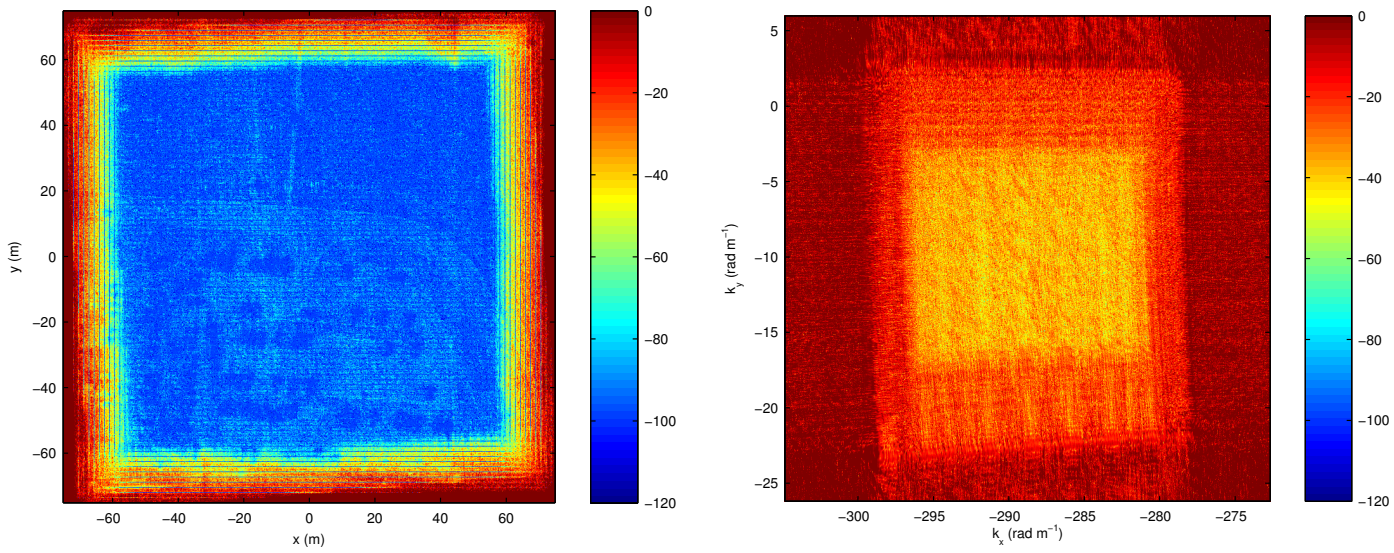
[2] J. McCorkle and M. Rofheart, "Order N^2 log(N) backprojector algorithm for focusing wide-angle wide-bandwidth arbitrary-motion synthetic aperture radar," in *Proc. SPIE*, vol. 2747, Apr. 1996, pp. 25–36.

[3] D. Wahl, D. Yocky, and C. Jakowatz Jr, "An implementation of a fast backprojection image formation algorithm for spotlight-mode SAR," in *Proc. SPIE*, vol. 6970, Apr. 2008.



(a) (b)

Fig. 3. Image formed using the angles $1-4^\circ$ of the eighth pass of the Gotcha Volumetric SAR Data Set. The image contains 768×768 pixels. The antenna beamwidths were limited to a 1.15° azimuth angle and a 0.57° polar angle. (a) was formed using the BP algorithm and (b) is its wavenumber spectrum.



(a) (b)

Fig. 4. Relative errors in the fast BP algorithms with respect to the BP algorithm. (a) is the relative error in the image that was formed using the fast decimation-in-image BP algorithm with three decomposition stages. (b) is the relative error in the wavenumber spectrum of the image that was formed using the fast decimation-in-phase-history BP algorithm with three decomposition stages.

[4] S. Basu and Y. Bresler, "O($N^2\log_2 N$) filtered backprojection reconstruction algorithm for tomography," vol. 9, no. 10, pp. 1760–1773, Oct. 2000.

[5] L. Demanet, M. Ferrara, N. Maxwell, J. Poulson, and L. Ying, "A butterfly algorithm for synthetic aperture radar imaging," *SIAM J. Imag. Sci.*, vol. 5, no. 1, pp. 203–243, Jan. 2012.

[6] L. Greengard and V. Rokhlin, "A fast algorithm for particle simulations," *J. Comput. Phys.*, vol. 73, no. 2, pp. 325–348, Dec. 1987.

[7] A. Dutt and V. Rokhlin, "Fast Fourier transforms for nonequispaced data, II," *Appl. Comp. Harm. Anal.*, vol. 2, no. 1, pp. 85–100, 1995.

[8] ——, "Fast Fourier transforms for nonequispaced data," *SIAM J. Sci. Comput.*, vol. 16, no. 6, pp. 1368–1393, 1993.

[9] C. Casteel Jr, L. Gorham, M. Minardi, S. Scarborough, K. Naidu, and U. Majumder, "A challenge problem for 2D/3D imaging of targets from a volumetric data set in an urban environment," in *Proc. SPIE*, vol. 6568, Apr. 2007.

[10] L. Greengard and J.-Y. Lee, "Accelerating the nonuniform fast Fourier transform," *SIAM Rev.*, vol. 46, no. 3, pp. 443–454, Jan. 2004.

[11] S. Kelly, G. Rilling, M. Davies, and B. Mulgrew, "Iterative image formation using fast (re/back)-projection for spotlight-mode SAR," in *Proc. IEEE Radar Conf.*, May 2011, pp. 835–840.