SAR image formation toolbox for MATLAB

LeRoy A. Gorham and Linda J. Moore

Air Force Research Laboratory, Sensors Directorate
2241 Avionics Circle, Bldg 620, WPAFB, OH 45433-7321

ABSTRACT

While many synthetic aperture radar (SAR) image formation techniques exist, two of the most intuitive methods for implementation by SAR novices are the matched filter and backprojection algorithms. The matched filter and (non-optimized) backprojection algorithms are undeniably computationally complex. However, the backprojection algorithm may be successfully employed for many SAR research endeavors not involving considerably large data sets and not requiring time-critical image formation. Execution of both image reconstruction algorithms in MATLAB is explicitly addressed. In particular, a manipulation of the backprojection imaging equations is supplied to show how common MATLAB functions, `ifft` and `interp1`, may be used for straight-forward SAR image formation. In addition, limits for scene size and pixel spacing are derived to aid in the selection of an appropriate imaging grid to avoid aliasing. Example SAR images generated though use of the backprojection algorithm are provided given four publicly available SAR datasets. Finally, MATLAB code for SAR image reconstruction using the matched filter and backprojection algorithms is provided.

Keywords: SAR, image formation, matched filter, backprojection algorithm

1. INTRODUCTION

Over the past four decades, numerous image formation algorithms have been developed with varying levels of complexity and accuracy. However, these algorithms have a steep learning curve for SAR novices. In this paper, we introduce two algorithms - matched filter and backprojection - implemented in MATLAB, which are very simple to understand. A benefit of matched filter image reconstruction is that the method offers optimization of the signal-to-noise ratio (SNR).[1] Use of the matched filter requires a hypothesis regarding a target's characteristics; these assumptions yield a specifically designed filter that will ideally match to the target. While the matched filter algorithm is intuitive and straight-forward, the generation of a 2D SAR image using this technique requires $\mathcal{O}(N^4)$ operations. For most practical situations, the computational complexity of the matched filter algorithm renders use of the method unrealistic. However, understanding of this fundamental imaging algorithm is extremely beneficial, especially for the SAR novice.

While several techniques exist for SAR image generation, two commonly employed approaches worth noting here are the polar format and backprojection algorithms. These algorithms differ in regards to the tradeoff between computational complexity and image quality.[2,3] The polar format algorithm[4,5] for SAR image reconstruction is similar to computer-aided tomography imaging techniques.[6,7,8] Polar format reconstruction employs a direct 2D Fourier transform, which provides the opportunity to take advantage of the extremely efficient fast Fourier transform. Polar format image reconstruction requires $\mathcal{O}(N^2 \log N)$ operations given an NxN SAR image. While more computationally effective than the backprojection algorithm, the polar format imaging method involves an approximation of the received signal phase. This approximation introduces errors into the final reconstructed SAR image. Several correction techniques have been suggested as a means of improving the accuracy of SAR images generated via polar format reconstruction.[9,10]

Another frequently used image formation technique is the backprojection algorithm.[7,11] A distinct advantage of the backprojection algorithm is the ability to form SAR images as phase history is collected, pulse by pulse, and to integrate newly obtained information into the SAR image as it becomes available. The backprojection algorithm is computationally expensive, however, as it requires $\mathcal{O}(N^3)$ operations for an NxN SAR image. Fortunately, the backprojection algorithm lends itself naturally to parallel processing. Backprojection has recently

Further author information: (E-mail: leroy.gorham@wpafb.af.mil and linda.moore2@wpafb.af.mil)

been implemented on graphics processing units (GPUs),[12,13] which provide inexpensive platforms for parallel computing.

Several methods have been proposed as a means of reducing the computational complexity of the backprojection algorithm while retaining the advantages of this image formation approach.[14,15,16,17] Many of these fast backprojection algorithms are flexible, providing manners in which image quality may be acceptably sacrificed in order to further improve data processing time. These image formation algorithms, several of which are favorable given wide bandwidths and large synthetic apertures, offer decreased computational complexities of $\mathcal{O}(N^{5/2})$[14] and $\mathcal{O}(N^2 \log N)$.[15,16,17] Non-optimized backprojection imaging may be impractical for exceptionally large data sets with time-critical image formation requirements; however, knowledge of this simple SAR imaging technique is undoubtedly valuable for many research endeavors.

The remainder of this paper is outlined as follows. In Section 2, we define our terminology and the SAR signal model. We also derive limits for scene size and pixel spacing to aid in selecting an imaging grid that avoids aliasing. In Section 3, we derive and describe the matched filter algorithm for image formation. In Section 4, we derive a backprojection algorithm, which is based on the matched filter algorithm. We describe how to utilize basic MATLAB functions like `ifft` and `interp1` to implement the algorithm. In Section 5, we apply the backprojection algorithm to four publicly available SAR datasets. Finally, MATLAB code for the two imaging algorithms, matched filter and backprojection, are supplied in Appendix A.1 and A.2.

The authors will be glad to provide copies of the MATLAB code used to generate the figures provided in this paper in response to requests via email.

2. SIGNAL MODEL

The following SAR signal model[18] is used for the development of MATLAB implementations of the matched filter and backprojection imaging algorithms. The SAR sensor travels along a flight path such that the antenna phase center has a three-dimensional spatial location denoted $\underline{r}_a(\tau)$ such that

$$\underline{r}_a(\tau) = [x_a(\tau), y_a(\tau), z_a(\tau)]^T \tag{1}$$

where τ denotes the synthetic aperture, or slow time, domain. All coordinates are defined with respect to a scene origin, which corresponds to the SAR motion compensation point. The distance from the antenna phase center to the origin, $\|\underline{r}_a(\tau)\|$, is denoted by

$$d_a(\tau) = \sqrt{x_a^2(\tau) + y_a^2(\tau) + z_a^2(\tau)}. \tag{2}$$

A target is specified at a location

$$\underline{r}(\tau) = [x(\tau), y(\tau), z(\tau)]^T. \tag{3}$$

In general, this target can have any arbitrary motion, but in this paper, we will assume that the target is stationary. Thus, the dependence on τ is dropped for the remainder of the derivation. The target has a radar cross section which is dependent on frequency and aspect angle. The distance from the antenna phase center to the target is denoted by

$$d_{a_0}(\tau) = \sqrt{(x_a(\tau) - x)^2 + (y_a(\tau) - y)^2 + (z_a(\tau) - z)^2}. \tag{4}$$

At periodic intervals, the radar transmits a pulse that reflects off scatterers in the scene. Some of the energy is then received by the radar. In a given synthetic aperture, there are N_p pulses used to form the image. The transmission time of each pulse is denoted by the sequence $\{\tau_n \ | n = 1, 2, ..., N_p\}$. The output of the receiver at a given time, τ_n, is a sequence of band-limited frequency samples delayed with respect to the time of pulse transmission by the round-trip time to the target. There are K frequency samples per pulse, and the associated frequency values are represented by the sequence $\{f_k | k = 1, 2, ..., K\}$. The receiver output from the target located at $\underline{r}$ is

$$S(f_k, \tau_n) = A(f_k, \tau_n) \exp\left(\frac{-j4\pi f_k \Delta R(\tau_n)}{c}\right). \tag{5}$$

The amplitude, $A(f_k, \tau_n)$, is related to the radar cross section of the target while the phase is dependent on the frequency of each sample and on the differential range, $\Delta R(\tau_n)$, given by

$$\Delta R(\tau_n) = d_{a_0}(\tau_n) - d_a(\tau_n). \tag{6}$$

Equation (5) assumes that each pulse is motion compensated such that a scatterer at the scene origin will have zero phase for all f_k and τ_n. The actual receiver output is thus the sum of the contributions of all scatterers in the scene.

The frequency samples, $\{f_k\}$, have a minimum value denoted by f_1, a median value denoted by f_c, a maximum value denoted by f_K, and a step size denoted by Δf. The frequency step size is inversely related to the maximum alias-free range extent of the image, W_r, by

$$W_r = \frac{c}{2\Delta f}. \tag{7}$$

Thus, the frequency step size is chosen to match the size of the scene to be imaged. The total bandwidth, B, of the received pulse is $B = (K-1)\Delta f$. The range resolution is thus

$$\delta_r = \frac{c}{2B} = \frac{c}{2(K-1)\Delta f}. \tag{8}$$

In a similar manner as above, the azimuth angle traversed during the synthetic aperture determines the cross-range resolution, and the azimuth angle from pulse to pulse determines the maximum alias-free cross-range extent of the image, W_x. Given an azimuth step size of $\Delta\theta$,

$$W_x = \frac{\lambda_{min}}{2\Delta\theta} \tag{9}$$

where λ_{min} is the minimum wavelength such that $\lambda_{min} = c/f_K$. The total azimuth angle, θ_a, traversed during the synthetic aperture is $\theta_a = (N_p - 1)\Delta\theta$. Thus, the cross-range resolution, δ_x, is given by

$$\delta_x = \frac{\lambda_c}{2\theta_a} = \frac{\lambda_c}{2(N_p-1)\Delta\theta} \tag{10}$$

where λ_c is the center wavelength such that $\lambda_c = c/f_c$.

To utilize the imaging algorithms described in this paper, one can select any arbitrary pixel locations. However, one should be careful when selecting these locations to avoid aliasing of the image or the frequency support. The pixel spacing should be finer than the resolution defined in Equations (8) and (10) and the overall scene size should be less than the maximum scene size defined in Equations (7) and (9).

3. MATCHED FILTER ALGORITHM

The most straightforward method for forming a SAR image is to perform a matched filter. One can build the matched filter to any kind of scatterer, but here we will assume an isotropic point scatterer. The received signal from a point scatterer at location $\underline{r}$ is given in Equation (5). An isotropic scatterer will have a constant amplitude and thus $A(f_k, \tau_n) = A_0$. Therefore, the matched filter response, denoted by $I(\underline{r})$, at location $\underline{r}$ is given by

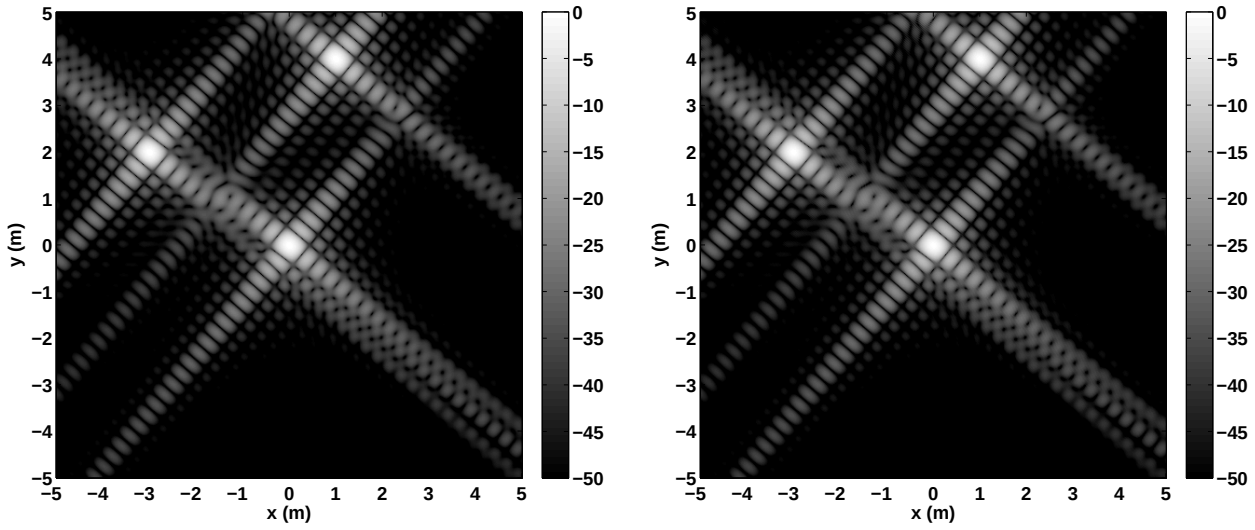
$$I(\underline{r}) = \frac{1}{N_p K} \sum_{n=1}^{N_p} \sum_{k=1}^{K} S(f_k, \tau_n) \exp\left(\frac{+j4\pi f_k \Delta R(\tau_n)}{c}\right) = A_0, \tag{11}$$

assuming a single scatterer in the scene.

To form an image using this method, Equation (11) is applied for each pixel in the image. This requires calculation of the differential range, $\Delta R(\tau_n)$, for every pixel for every pulse. The algorithm has a computational complexity of $\mathcal{O}(N^4)$ for 2D images, which makes it impractical for most applications. However, Equation (11) forms the basis for the derivation of the backprojection algorithm in Section 4. MATLAB code for the matched filter algorithm is provided in Appendix A.1.

MATLAB	Type	Symbol	Description	Units
`data.deltaF`	scalar	Δf	Frequency step size	Hz
`data.minF`	N_p vector	f_1	Minimum frequency for every pulse	Hz
`data.Nfft`	scalar	$N_{\texttt{fft}}$	Length of FFT (backprojection only)	
`data.x_mat`	Arbitrary	$\underline{r}$	X coordinate of every pixel in the image	m
`data.y_mat`	Arbitrary	$\underline{r}$	Y coordinate of every pixel in the image	m
`data.z_mat`	Arbitrary	$\underline{r}$	Z coordinate of every pixel in the image	m
`data.AntX`	N_p vector	$\underline{r}_a$	Antenna location for every pulse	m
`data.AntY`	N_p vector	$\underline{r}_a$	Antenna location for every pulse	m
`data.AntZ`	N_p vector	$\underline{r}_a$	Antenna location for every pulse	m
`data.R0`	N_p vector	$d_a(\tau_n)$	Distance to the mo-comp point	m
`data.phdata`	(KxN_p) array	$S(f_k, \tau_n)$	Complex phase history data, Eq. (5)	

Table 1. Necessary fields in the MATLAB data structure for image formation.



(a) matched filter image

(b) backprojection image

Figure 1. Comparison of SAR images (in dB) of 3 point targets, located at (0,0,0), (-3,2,0), and (1,4,0), generated using the (a) matched filter and (b) backprojection algorithms.

In order to use the MATLAB function, `mfBasic`, the inputs are provided in a structure called `data`. The fields of this structure are defined in Table 1. The image pixel locations, $\underline{r}$, are specified in the pixel location matrices `data.x_mat`, `data.y_mat` and `data.z_mat` which contain the (x, y, z) locations of each pixel. The output image is stored in `data.im_final`, which has the same dimensions as the pixel location matrices.

While these pixels can have arbitrary positions, it is often desirable to form an image onto a regular 2D or 3D grid. In this case, the MATLAB function `meshgrid` is useful to build the pixel location matrices. In all the image examples provided in this paper, `meshgrid` was used to build `data.x_mat` and `data.y_mat`, while `data.z_mat` was filled with zeros. If a Digital Elevation Map (DEM) is available, `data.z_mat` can be filled with interpolated height values for every pixel to improve image quality.

Figure 1(a) shows an example of an image formed using the matched filter algorithm. Phase history data was simulated for 3 point targets using Equation (5) where $A(f_k, \tau_n) = 1$. Here, $N_p = 128$ pulses were simulated with $K = 512$ frequency samples per pulse, a center frequency of 10 GHz and a 600 MHz bandwidth. A circular flight path was used with a 30 degree depression angle and a slant range of 10 km. A 3 degree integration angle was used with a center azimuth angle of 50 degrees. The scene extent was 10 m x 10 m with 2 cm pixel spacing in each dimension.

4. BACKPROJECTION ALGORITHM

The backprojection algorithm offers an intuitive imaging technique for SAR novices. While the backprojection algorithm is computationally expensive, its implementation is not unreasonable for many SAR research ventures. In this section, backprojection imaging equations are manipulated in order to illustrate how the algorithm can be executed through use of common MATLAB functions.

4.1 Derivation of Efficient Calculation of Range Profiles

The matched filter response shown in Equation (11) can be used to compute the target response at a discrete range bin, m. Given SAR phase history, $S(f_k, \tau_n)$, collected by N_p pulses over a range of K frequencies, the range profile at range bin m given a received pulse at slow time τ_n is

$$s(m, \tau_n) = \sum_{k=1}^{K} S(f_k, \tau_n) \exp\left(\frac{+j4\pi f_k \Delta R(m, \tau_n)}{c}\right). \tag{12}$$

By substituting the frequency values $f_k = (k-1)\Delta f + f_1$ into Equation (12), the range profile may be rewritten as

$$\begin{aligned} s(m, \tau_n) &= \sum_{k=1}^{K} S(f_k, \tau_n) \exp\left(\frac{+j4\pi((k-1)\Delta f + f_1)\Delta R(m, \tau_n)}{c}\right) \\ &= \sum_{k=1}^{K} S(f_k, \tau_n) \exp\left(\frac{+j4\pi\Delta f \Delta R(m, \tau_n)(k-1)}{c} + \frac{+j4\pi f_1 \Delta R(m, \tau_n)}{c}\right) \\ &= \sum_{k=1}^{K} S(f_k, \tau_n) \exp[\Phi(\Delta R(m, \tau_n)) \cdot (k-1)] \exp\left(\frac{+j4\pi f_1 \Delta R(m, \tau_n)}{c}\right) \end{aligned} \tag{13}$$

where phase function $\Phi(\Delta R(m, \tau_n)) = (+j4\pi\Delta f \Delta R(m, \tau_n))/c$.

In order to implement the backprojection algorithm in MATLAB, Equation (13) must be rewritten in terms of MATLAB's inverse discrete Fourier transform (`ifft`) function. In MATLAB, the definitions of the discrete Fourier transforms between $X(k)$ and $x(m)$ are given as[19]

$$\begin{aligned} X(k) &= \sum_{m=1}^{M} x(m) \cdot \omega_K^{(m-1)(k-1)} \\ x(m) &= \frac{1}{K} \sum_{k=1}^{K} X(k) \cdot \omega_K^{-(m-1)(k-1)} \end{aligned} \tag{14}$$

where $\omega_K = \exp(-j2\pi/K)$. Therefore, the inverse discrete Fourier transform of $X(k)$ is given as

$$x(m) = \texttt{ifft}(X(k)) = \frac{1}{K} \sum_{k=1}^{K} X(k) \exp\left(\frac{+j2\pi(m-1)}{K} \cdot (k-1)\right) = \frac{1}{K} \sum_{k=1}^{K} X(k) \exp\left(\Theta(m) \cdot (k-1)\right) \tag{15}$$

where $\Theta(m) = (j2\pi(m-1))/K$. In order to write Equation (13) in terms of MATLAB's `ifft` function, the previously defined phase function, $\Phi(\Delta R(m, \tau_n))$, from Equation(13), must equal $\Theta(m)$ from Equation (15). To satisfy this requirement, the following equality must hold:

$$\Delta R(m, \tau_n) = \frac{(m-1)}{K} \cdot \frac{c}{2\Delta f} = \frac{(m-1)}{K} \cdot W_r. \tag{16}$$

Note that $\Delta R(m, \tau_n)$ represents a sampling across range and the maximum unambiguous range occurs when $(m-1) = K$ (recall Equation (7)).

Inserting the result from Equation (16) into Equation (13) yields

$$\begin{aligned} s(m,\tau_n) &= \sum_{k=1}^{K} S(f_k,\tau_n)\exp\left(\frac{+j4\pi\Delta f(k-1)}{c}\cdot\Delta R(m,\tau_n)\right)\cdot\exp\left(\frac{+j4\pi f_1\Delta R(m,\tau_n)}{c}\right) \\ &= \sum_{k=1}^{K} S(f_k,\tau_n)\exp\left(\frac{+j4\pi\Delta f(k-1)}{c}\cdot\left(\frac{m-1}{K}\cdot\frac{c}{2\Delta f}\right)\right)\cdot\exp\left(\frac{+j4\pi f_1\Delta R(m,\tau_n)}{c}\right) \\ &= \sum_{k=1}^{K} S(f_k,\tau_n)\exp\left(\frac{+j2\pi(m-1)}{K}\cdot(k-1)\right)\cdot\exp\left(\frac{+j4\pi f_1\Delta R(m,\tau_n)}{c}\right). \end{aligned} \tag{17}$$

Finally, recalling the expression for the inverse discrete Fourier transform from Equation (15), the range profile may be expressed in terms of MATLAB's `ifft`.

$$s(m,\tau_n) = K\cdot\texttt{ifft}(S(f_k,\tau_n))\cdot\exp\left(\frac{+j4\pi f_1\Delta R(m,\tau_n)}{c}\right) \tag{18}$$

The constant K appears in Equation (18) because by definition, the inverse discrete Fourier transform contains a scaling factor of $1/K$. In addition, it's important to note that the MATLAB implementation of the inverse discrete Fourier transform assumes that K is even. However, for values of K that are powers of 2, the computational complexity of the inverse discrete Fourier transform is optimized. By inserting the expression for $\Delta R(m,\tau_n)$ from Equation (16), an alternative expression for the implementation of a range profile in MATLAB is given by

$$s(m,\tau_n) = K\cdot\texttt{ifft}(S(f_k,\tau_n))\cdot\exp\left(\frac{j2\pi f_1(m-1)}{K\Delta f}\right). \tag{19}$$

One final consideration must be addressed. Each pulse is motion compensated such that a scatterer at the scene origin has zero phase and will appear in the zero frequency bin in the range profile. This bin corresponds to the center of the range profile computed in Equation (19). By default, the function `ifft` computes the values from $1 \le m \le K$ where $m = 1$ corresponds to the zero frequency bin. To put $m = 1$ at the center of the range profile, the command `fftshift` is applied to the output of the `ifft` command. This operation ensures that the output is correctly ordered following the inverse discrete Fourier transform operation and that the zero frequency component corresponds to the center of the output vector.

As a result, range index, m, acquires values between $-K/2+1$ and $K/2$. Therefore, $\Delta R(m,\tau_n)$ of Equation (16) has values between $-W_r/2$ and $W_r/2 - W_r/K$ and Equation (19) becomes

$$s(m,\tau_n) = K\cdot\texttt{fftshift}\{\texttt{ifft}(S(f_k,\tau_n))\}\cdot\exp\left(\frac{j2\pi f_1(m-1)}{K\Delta f}\right). \tag{20}$$

4.2 Image Formation Process

For implementation of the backprojection imaging algorithm in MATLAB, Equation (6) is used to compute the differential range, $\Delta R(\tau_n)$, for each pixel, for each pulse, where the pixel (x,y,z) coordinates are inserted into Equation (4). In order to use Equation (20) to form a SAR image, an interpolation step is required due to the fact that the discrete values of $\Delta R(m,\tau_n)$ do not correspond exactly to the $\Delta R(\tau_n)$ values calculated for every pixel. There are several methods for performing this interpolation,[20,11] but here we will simply use the MATLAB `interp1` function to implement linear interpolation.

Since $s(m,\tau_n)$ is a band-limited signal, the ideal interpolator is a sinc interpolator. This can be approximated by zero-padding the `ifft` computation and then performing linear interpolation on the inverse discrete Fourier transform output. A good rule of thumb is that the length of the `ifft`, denoted $N_{\texttt{fft}}$, should be 10 times the length of the data, K. Also, the `ifft` function is most efficient when $N_{\texttt{fft}}$ is a power of 2. In the backprojection algorithm, $N_{\texttt{fft}}$ is provided as an input to the imaging function, as shown in Table 1.

The first step in the image formation process is to implement Equation (20) for every pulse, zero-padding the data such that the length of the `ifft` is $N_{\texttt{fft}}$. Thus,

$$s(m,\tau_n) = N_{\texttt{fft}} \cdot \texttt{fftshift}\{\texttt{ifft}(S(f_k,\tau_n))\} \cdot \exp\left(\frac{j2\pi f_1(m-1)}{N_{\texttt{fft}}\Delta f}\right). \tag{21}$$

where $S(f_k,\tau_n) = 0$ for all $k > K$.

To find the image response for a pixel at location $\underline{r}$, given pulse n, $\Delta R(\tau_n)$ is calculated and used to find an interpolated value of $s(m,\tau_n)$. This is denoted as $s_{int}(\underline{r},\tau_n)$. The final image response, $I(\underline{r})$, is simply the summation of these values for every pulse.

$$I(\underline{r}) = \sum_{n=1}^{N_p} s_{int}(\underline{r},\tau_n). \tag{22}$$

Figure 1(b) shows an example of an image formed using the backprojection algorithm. The image was generated using the same data and equivalent pixel locations as those employed in Figure 1(a).

5. IMAGE EXAMPLES

In the last few years, the Air Force Research Laboratory (AFRL) has released several datasets of both synthetic and measured SAR data. In this section, the backprojection algorithm introduced in Section 4 is applied to data from the Backhoe Data Dome, the 2D/3D Volumetric Challenge Problem, and the SAR-based Ground Moving Target Indicator (GMTI) Challenge Problem. In addition, the imaging code was implemented on the Civilian Vehicle Radar Data Domes synthetic data set from The Ohio State University.

5.1 Backhoe Data Dome

In 2004, AFRL released a synthetically generated data dome of a backhoe target.[21] The scattered field data for the backhoe was computed on a 2π-steradian data dome over the target for a frequency bandwidth of 5.9 GHz at a center frequency of 10 GHz.

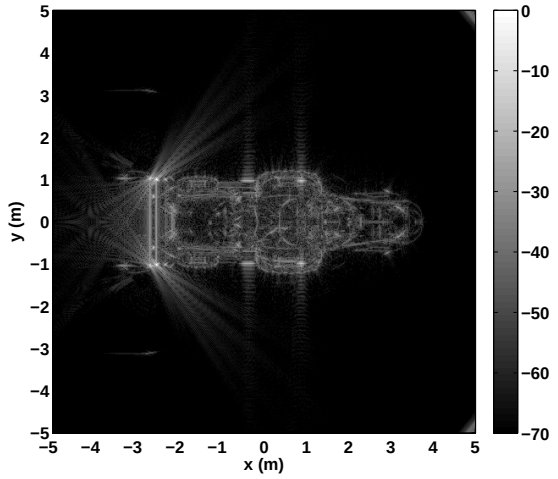


Figure 2. Backprojection image (in dB) using the Backhoe Data Dome synthetic dataset.

The backprojection algorithm was performed on this dataset and a resulting image is given in Figure 2. The data imaged consisted of all 360° aspect angles from a single elevation of 10° using VV polarization. The full 5.9 GHz bandwidth was used to form the image. The scene extent is 10 m x 10 m with 2 cm pixel spacing, resulting in a 501 x 501 pixel image. No windowing was applied to the data before imaging. The calculated maximum scene size is $W_r = 12.97$ m; $W_x = 9.27$ m.

5.2 2D/3D Volumetric Challenge Problem

In 2007, AFRL released a challenge problem for 2D/3D imaging of targets from a volumetric data set in an urban environment.[22] The data was collected of a scene consisting of numerous civilian vehicles and calibration targets.

The backprojection algorithm was performed on this dataset and a resulting image is given in Figure 3. The data imaged consisted of Pass 1 with HH polarization. An integration angle, $\Delta\theta$, of 4° centered at 40° azimuth was used. The scene extent is 100 m x 100 m with 20 cm pixel spacing, resulting in a 501 x 501 pixel image. No windowing was applied to the data before imaging. The calculated maximum scene size is $W_r = 101.87$ m; $W_x = 108.41$ m and the calculated resolution is $\delta r = 0.24$ m; $\delta x = 0.23$ m.

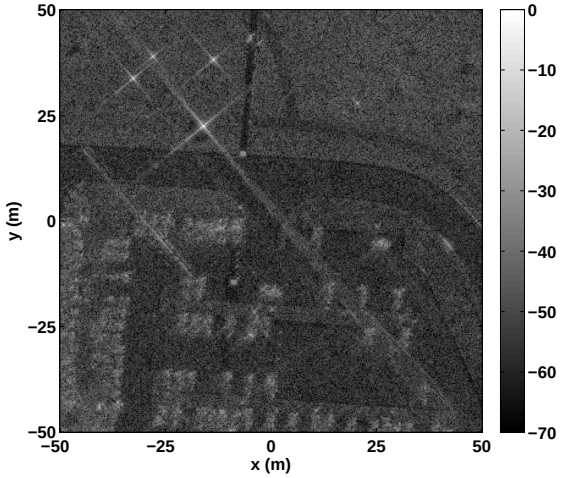


Figure 3. Backprojection image (in dB) using the 2D/3D Volumetric Challenge Problem dataset.

5.3 SAR-based GMTI Challenge Problem

In 2009, AFRL released a challenge problem for SAR-based GMTI in urban environments.[23] This data consists of a 71-second portion of phase history data from a radar operating in circular SAR mode. The data has been range-gated around the known location of a moving vehicle, severely limiting the range swath of the data. However, during the first 25 seconds of the scenario, the vehicle is waiting in line to cross a busy intersection.

The backprojection algorithm was performed on this dataset and a resulting image is provided in Figure 4. The scene extent is 200 m x 200 m with 20 cm pixel spacing, resulting in a 1001 x 1001 pixel image. The first 8000 pulses (almost 4 seconds) were used, and no windowing was applied to the data before imaging. The calculated maximum scene size is $W_r = 89.94$ m; $W_x = 2,032.41$ m, and the calculated resolution is $\delta r = 0.23$ m; $\delta x = 0.25$ m. The relatively small range extent accounts for the black areas in the image.

5.4 Civilian Vehicle Radar Data Domes

In 2010, The Ohio State University released a set of synthetically generated data domes of civilian targets.[24] The scattered field data for the targets was computed for full azimuth coverage of elevation angles from 30° to 60°. The data is full polarimetric with a frequency bandwidth of 5.35 GHz at a center frequency of 9.6 GHz.

The backprojection algorithm was implemented on the 1993 Jeep target, and a resulting image is shown in Figure 5. The data imaged consisted of all 360° aspect angles from a single elevation of 30° using VV polarization. The full bandwidth was used to form the image. The scene extent is 10 m x 10 m with 2 cm pixel spacing, resulting in a 501 x 501 pixel image. No windowing was applied to the data before imaging. The calculated maximum scene size is $W_r = 14.30$ m; $W_x = 11.18$ m.

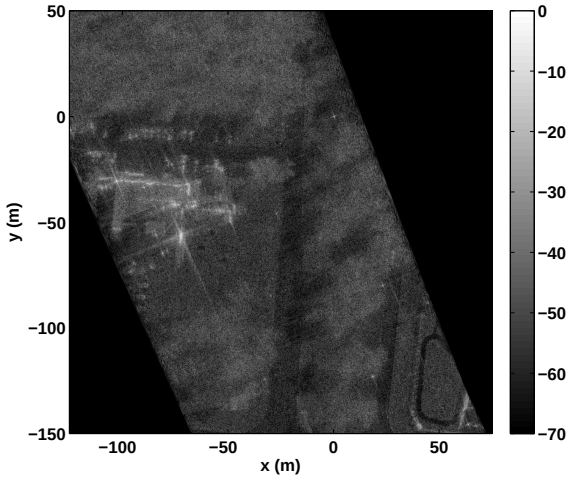


Figure 4. Backprojection image (in dB) using the SAR-based GMTI dataset.

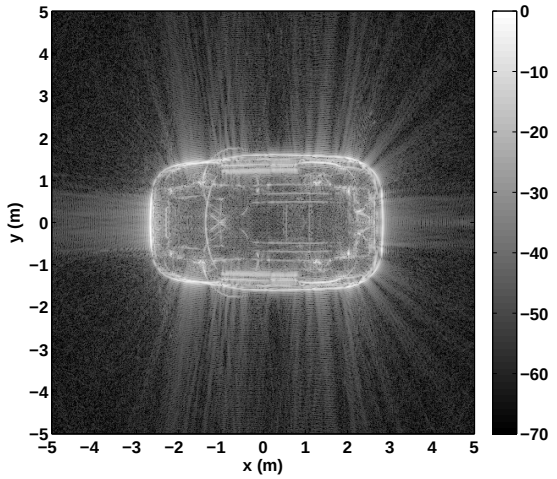


Figure 5. Backprojection image (in dB) using the Civilian Vehicle Radar Data Domes synthetic dataset.

6. CONCLUSIONS

In this paper, we developed two simple SAR imaging algorithms: a matched filter algorithm and a backprojection algorithm. The backprojection algorithm was derived from the matched filter algorithm by utilizing a fast Fourier transform to compute range profiles. Both algorithms were implemented in MATLAB, and image examples were provided using four publicly available SAR datasets.

APPENDIX A. MATLAB CODE

A.1 Matched Filter Algorithm

```
function data = mfBasic(data)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This function performs a matched filter operation.  The following    %
% fields need to be populated:                                         %
%                                                                      %
% data.deltaF:  Step size of frequency data (Hz)                       %
% data.minF:  Vector containing the start frequency of each pulse (Hz) %
% data.x_mat:  The x-position of each pixel (m)                        %
% data.y_mat:  The y-position of each pixel (m)                        %
% data.z_mat:  The z-position of each pixel (m)                        %
% data.AntX:  The x-position of the sensor at each pulse (m)           %
% data.AntY:  The y-position of the sensor at each pulse (m)           %
% data.AntZ:  The z-position of the sensor at each pulse (m)           %
% data.R0:  The range to scene center (m)                              %
% data.phdata:  Phase history data (frequency domain)                  %
%               Fast time in rows, slow time in columns                %
%                                                                      %
% The output is:                                                       %
% data.im_final:  The complex image value at each pixel                %
%                                                                      %
% Written by LeRoy Gorham, Air Force Research Laboratory, WPAFB, OH    %
% Email:  leroy.gorham@wpafb.af.mil                                    %
% Date Released:  8 Apr 2010                                           %
%                                                                      %
% Gorham, L.A. and Moore, L.J., "SAR image formation toolbox for       %
%   MATLAB,"  Algorithms for Synthetic Aperture Radar Imagery XVII     %
%   7669, SPIE (2010).                                                 %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Define speed of light (m/s)
c = 299792458;

% Determine the size of the phase history data
data.K = size(data.phdata,1);  % The number of frequency bins per pulse
data.Np = size(data.phdata,2); % The number of pulses

% Determine the azimuth angles of the image pulses (radians)
data.AntAz = unwrap(atan2(data.AntY,data.AntX));

% Determine the average azimuth angle step size (radians)
data.deltaAz = abs(mean(diff(data.AntAz)));

% Determine the total azimuth angle of the aperture (radians)
data.totalAz = max(data.AntAz) - min(data.AntAz);

% Determine the maximum scene size of the image (m)
data.maxWr = c/(2*data.deltaF);
data.maxWx = c/(2*data.deltaAz*mean(data.minF));

% Determine the resolution of the image (m)
data.dr = c/(2*data.deltaF*data.K);
data.dx = c/(2*data.totalAz*mean(data.minF));

% Display maximum scene size and resolution
fprintf('Maximum Scene Size:  %.2f m range, %.2f m cross-range\n',data.maxWr,data.maxWx);
fprintf('Resolution:  %.2fm range, %.2f m cross-range\n',data.dr,data.dx);

% Initialize the image with all zero values
data.im_final = zeros(size(data.x_mat));

% Set up a vector to keep execution times for each pulse (sec)
t = zeros(1,data.Np);

% Loop through every pulse
for ii = 1:data.Np

    % Display status of the imaging process
    if ii > 1
        t_sofar = sum(t(1:(ii-1)));
        t_est = (t_sofar*data.Np/(ii-1)-t_sofar)/60;
        fprintf('Pulse %d of %d, %.02f minutes remaining\n',ii,data.Np,t_est);
    else
```

```
        fprintf('Pulse %d of %d\n',ii,data.Np);
    end
    tic

    % Calculate differential range for each pixel in the image (m)
    dR = sqrt((data.AntX(ii)-data.x_mat).^2 + ...
        (data.AntY(ii)-data.y_mat).^2 + ...
        (data.AntZ(ii)-data.z_mat).^2) - data.R0(ii);

    % Calculate the frequency of each sample in the pulse (Hz)
    freq = data.minF(ii) + (0:(data.K-1)) * data.deltaF;

    % Perform the Matched Filter operation
    for jj = 1:data.K
        data.im_final = data.im_final + data.phdata(jj,ii) * exp(1i*4*pi*freq(jj)/c*dR);
    end

    % Determine the execution time for this pulse
    t(ii) = toc;
end

return
```

A.2 Backprojection Algorithm

```
function data = bpBasic(data)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This function performs a basic Backprojection operation.  The        %
% following fields need to be populated:                               %
%                                                                      %
% data.Nfft:  Size of the FFT to form the range profile                %
% data.deltaF:  Step size of frequency data (Hz)                       %
% data.minF:  Vector containing the start frequency of each pulse (Hz) %
% data.x_mat:  The x-position of each pixel (m)                        %
% data.y_mat:  The y-position of each pixel (m)                        %
% data.z_mat:  The z-position of each pixel (m)                        %
% data.AntX:  The x-position of the sensor at each pulse (m)           %
% data.AntY:  The y-position of the sensor at each pulse (m)           %
% data.AntZ:  The z-position of the sensor at each pulse (m)           %
% data.R0:  The range to scene center (m)                              %
% data.phdata:  Phase history data (frequency domain)                  %
%               Fast time in rows, slow time in columns                %
%                                                                      %
% The output is:                                                       %
% data.im_final:  The complex image value at each pixel                %
%                                                                      %
% Written by LeRoy Gorham, Air Force Research Laboratory, WPAFB, OH    %
% Email:  leroy.gorham@wpafb.af.mil                                    %
% Date Released:  8 Apr 2010                                           %
%                                                                      %
% Gorham, L.A. and Moore, L.J., "SAR image formation toolbox for       %
%   MATLAB,"  Algorithms for Synthetic Aperture Radar Imagery XVII     %
%   7669, SPIE (2010).                                                 %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Define speed of light (m/s)
c = 299792458;

% Determine the size of the phase history data
data.K = size(data.phdata,1);  % The number of frequency bins per pulse
data.Np = size(data.phdata,2); % The number of pulses

% Determine the azimuth angles of the image pulses (radians)
data.AntAz = unwrap(atan2(data.AntY,data.AntX));

% Determine the average azimuth angle step size (radians)
data.deltaAz = abs(mean(diff(data.AntAz)));

% Determine the total azimuth angle of the aperture (radians)
data.totalAz = max(data.AntAz) - min(data.AntAz);

% Determine the maximum scene size of the image (m)
data.maxWr = c/(2*data.deltaF);
data.maxWx = c/(2*data.deltaAz*mean(data.minF));

```

```
% Determine the resolution of the image (m)
data.dr = c/(2*data.deltaF*data.K);
data.dx = c/(2*data.totalAz*mean(data.minF));

% Display maximum scene size and resolution
fprintf('Maximum Scene Size:  %.2f m range, %.2f m cross-range\n',data.maxWr,data.maxWx);
fprintf('Resolution:  %.2fm range, %.2f m cross-range\n',data.dr,data.dx);

% Calculate the range to every bin in the range profile (m)
data.r_vec = linspace(-data.Nfft/2,data.Nfft/2-1,data.Nfft)*data.maxWr/data.Nfft;

% Initialize the image with all zero values
data.im_final = zeros(size(data.x_mat));

% Set up a vector to keep execution times for each pulse (sec)
t = zeros(1,data.Np);

% Loop through every pulse
for ii = 1:data.Np

    % Display status of the imaging process
    if ii > 1
        t_sofar = sum(t(1:(ii-1)));
        t_est = (t_sofar*data.Np/(ii-1)-t_sofar)/60;
        fprintf('Pulse %d of %d, %.02f minutes remaining\n',ii,data.Np,t_est);
    else
        fprintf('Pulse %d of %d\n',ii,data.Np);
    end
    tic

    % Form the range profile with zero padding added
    rc = fftshift(ifft(data.phdata(:,ii),data.Nfft));

    % Calculate differential range for each pixel in the image (m)
    dR = sqrt((data.AntX(ii)-data.x_mat).^2 + ...
        (data.AntY(ii)-data.y_mat).^2 + ...
        (data.AntZ(ii)-data.z_mat).^2) - data.R0(ii);

    % Calculate phase correction for image
    phCorr = exp(1i*4*pi*data.minF(ii)/c*dR);

    % Determine which pixels fall within the range swath
    I = find(and(dR > min(data.r_vec), dR < max(data.r_vec)));

    % Update the image using linear interpolation
    data.im_final(I) = data.im_final(I) + interp1(data.r_vec,rc,dR(I),'linear') .* phCorr(I);

    % Determine the execution time for this pulse
    t(ii) = toc;
end

return
```

REFERENCES

[1] Soumekh, M., [*Synthetic aperture radar signal processing with MATLAB algorithms*], Wiley-Interscience (1999).

[2] Jakowatz Jr, C., Wahl, D., Yocky, D., Bray, B., Bow Jr, W., and Richards, J., "Comparison of algorithms for use in real-time spotlight-mode SAR image formation," in [*Proceedings of SPIE*], **5427**, 108 (2004).

[3] Jakowatz Jr, C. and Doren, N., "Comparison of polar formatting and back-projection algorithms for spotlight-mode SAR image formation," in [*Proceedings of SPIE*], **6237**, 62370H (2006).

[4] Carrara, W., Goodman, R., and Majewski, R., [*Spotlight Synthetic Aperture Radar - Signal Processing Algorithms*], Norwood, MA: Artech House (1995).

[5] Jakowatz, C., Wahl, D., Eichel, P., Ghiglia, D., and Thompson, P., [*Spotlight-mode synthetic aperture radar: a signal processing approach*], Kluwer Academic Pub (1996).

[6] Stark, H., Woods, J., Paul, I., and Hingorani, R., "An investigation of computerized tomography by direct fourier inversion and optimum interpolation," *IEEE Transactions on Biomedical Engineering* , 496–505 (1981).

[7] Munson Jr, D., O'Brien, J., and Jenkins, W., "A tomographic formulation of spotlight-mode synthetic aperture radar," *Proceedings of the IEEE* **71**(8), 917–925 (1983).

[8] Gorham, L. A., Rigling, B. D., and Zelnio, E. G., "A comparison between imaging radar and medical imaging polar format algorithm implementations," in [*Proceedings of SPIE*], **6568**, 65680K (2007).

[9] Doren, N., Jakowatz Jr, C., Wahl, D., and Thompson, P., "General formulation for wavefront curvature correction in polar-formatted spotlight-mode SAR images using space-variant post-filtering," in [*Proceedings of the 1997 International Conference on Image Processing (ICIP'97) 3-Volume Set-Volume 1-Volume 1*], IEEE Computer Society Washington, DC, USA (1997).

[10] Preiss, M., Gray, D., and Stacy, N., "Space Variant Filtering of Polar Format Spotlight SAR Images for Wavefront Curvature Correction and Interferometric Processing," in [*IGARSS 2002: IEEE International Geoscience and Remote Sensing Symposium (24th: 2002: Toronto, Ontario)*], IEEE: Institute of Electrical and Electronics Engineers (2002).

[11] Desai, M. and Jenkins, W., "Convolution backprojection image reconstruction for spotlight mode synthetic aperture radar," *IEEE Transactions on Image Processing* **1**(4), 505–517 (1992).

[12] Hartley, T., Fasih, A., Berdanier, C., Özgüner, F., and Çatalyürek, U., "Investigating the use of GPU-accelerated nodes for SAR image formation," in [*Cluster Computing and Workshops, 2009. CLUSTER '09. IEEE International Conference on*], 1–8 (Aug 31–Sep 4 2009).

[13] Rogan, A. and Carande, R., "Improving the fast backprojection algorithm through massive parallelizations," *Radar Sensor Technology XIV* **7669**(1), SPIE (2010).

[14] Yegulalp, A., "Fast backprojection algorithm for synthetic aperture radar," in [*Radar Conference, 1999. The Record of the 1999 IEEE*], 60–65 (1999).

[15] McCorkle, J. and Rofheart, M., "An order N^2log(N) backprojector algorithm for focusing wide-angle wide-bandwidth arbitrary-motion synthetic aperture radar," in [*Proc. of SPIE Conference on Radar Sensor Technology*], **2747**, 25–36 (1996).

[16] Ulander, L., Hellsten, H., and Stenstrom, G., "Synthetic-aperture radar processing using fast factorized back-projection," *IEEE Transactions on Aerospace and electronic systems* **39**(3), 760–776 (2003).

[17] Wahl, D. E., Yocky, D. A., and Jakowatz Jr., C. V., "An implementation of a fast backprojection image formation algorithm for spotlight-mode SAR," in [*Proceedings of SPIE*], **6970**, 69700H (2008).

[18] Rigling, B. and Moses, R., "Taylor expansion of the differential range for monostatic SAR," *IEEE Transactions on Aerospace and Electronic Systems* **41**(1), 60–64 (2005).

[19] The Mathworks, Inc, *MATLAB Documentation: fft - Discrete Fourier Transform.* Version 7.9.0.529 (R2009b).

[20] Crochiere, R. and Rabiner, L., "Interpolation and decimation of digital signals - a tutorial review," *Proceedings of the IEEE* **69**(3), 300–331 (1981).

[21] Naidu, K. and Lin, L., "Data dome: full k-space sampling data for high-frequency radar research," *Algorithms for Synthetic Aperture Radar Imagery XI* **5427**(1), 200–207, SPIE (2004).

[22] Casteel Jr, C. H., Gorham, L. A., Minardi, M. J., Scarborough, S. M., Naidu, K. D., and Majumder, U. K., "A challenge problem for 2D/3D imaging of targets from a volumetric data set in an urban environment," *Algorithms for Synthetic Aperture Radar Imagery XIV* **6568**(1), 65680D, SPIE (2007).

[23] Scarborough, S. M., Casteel Jr, C. H., Gorham, L., Minardi, M. J., Majumder, U. K., Judge, M. G., Zelnio, E., Bryant, M., Nichols, H., and Page, D., "A challenge problem for SAR-based GMTI in urban environments," *Algorithms for Synthetic Aperture Radar Imagery XVI* **7337**(1), 73370G, SPIE (2009).

[24] Dungan, K. E., Austin, C., Nehrbass, J., and Potter, L. C., "Civilian vehicle radar data domes," *Algorithms for Synthetic Aperture Radar Imagery XVII* **7699**(1), SPIE (2010).