PERFECT Benchmark Suite Manual

Kevin Barker (PNNL)
Thomas Benson (GTRI)
Dan Campbell (GTRI)
David Ediger (GTRI)
Roberto Gioiosa (PNNL)
Adolfy Hoisie (PNNL)
Darren Kerbyson (PNNL)
Joseph Manzano (PNNL)
Andres Marquez (PNNL)
Leon Song (PNNL)
Nathan Tallent (PNNL)
Antonino Tumeo (PNNL)

Version 0.0.5 ($Revision : 8305$)

Contents

Chapter 1

PERFECT Application 1 (PA1)

1.1 Kernel 1 – 2D Convolution

1.1.1 Motivation

Convolution expresses the overlap of two functions f and g as g is shifted over f. In image and signal processing, the function f is the input image or signal and g is referred to as the filter. The filter is shifted over and applied to each pixel resulting in an output image of the same size. Boundary pixels must be handled carefully, usually by zero-padding the input.

The Gaussian filter is a filter whose impulse response is a Gaussian function. When applied in two dimensions, each pixel in the output image represents a weighted average of pixels in the neighborhood, where the weighting diminishes in concentric circles around the focal point. The Gaussian filter can be used to reduce noise in images.

1.1.2 Description

The input image uses 32-bit unsigned pixels with a dynamic range of 16 bits per pixel. There are three sizes of input images: small (640x480), medium, (1920x1080), and large (3840x2160). The image width is M pixels and the height is N pixels. The input image is convolved with a 9x9 Gaussian filter to produce an output image with the same number of rows and columns. The image is zero-padded so that the number of output pixels is equal to the number of input pixels after convolution. This process is repeated for each of 30 input frames.

The Gaussian 2D convolution is separable into two one-dimensional convolutions. The one-dimensional filter is defined as $h[x]$.

$$h[x] = \begin{bmatrix} 1 & 3 & 4 & 5 & 6 & 5 & 4 & 3 & 1 \end{bmatrix} \tag{1.1}$$

Multiplying by the transpose gives the complete 2D convolution filter $g[x,y]$.

$$g[x,y] = h' \cdot h \tag{1.2}$$

$$g[x,y] = \begin{bmatrix}
1 & 3 & 4 & 5 & 6 & 5 & 4 & 3 & 1 \\
3 & 9 & 12 & 15 & 18 & 15 & 12 & 9 & 3 \\
4 & 12 & 16 & 20 & 24 & 20 & 16 & 12 & 4 \\
5 & 15 & 20 & 25 & 30 & 25 & 20 & 15 & 5 \\
6 & 18 & 24 & 30 & 36 & 30 & 24 & 18 & 6 \\
5 & 15 & 20 & 25 & 30 & 25 & 20 & 15 & 5 \\
4 & 12 & 16 & 20 & 24 & 20 & 16 & 12 & 4 \\
3 & 9 & 12 & 15 & 18 & 15 & 12 & 9 & 3 \\
1 & 3 & 4 & 5 & 6 & 5 & 4 & 3 & 1
\end{bmatrix} \tag{1.3}$$

For the input image $f[x,y]$, the output of the benchmark is the convolution $f[x,y] * g[x,y]$ with normalized pixel intensities.

$$f[x,y] * g[x,y] = \frac{1}{1024} \sum_{n_1=-4}^{4} \sum_{n_2=-4}^{4} f[n_1,n_2] \cdot g[x-n_1, y-n_2] \tag{1.4}$$

1.1.3 Specification

Input Data: 30 frames; 16 bits per pixel; small (640x480), medium (1920x1080), large (3840x2160)

Output Data: 30 frames, 16 bits per pixel, with the 9x9 Gaussian $g[x,y]$ in Eqn. 1.3 applied

Key Operations: Perform a 2D convolution of the image input with the 9x9 filter, normalizing pixel intensities by 1/1024

1.1.4 Assessment

Correctness Assessment

The 2D convolution kernel should be compared with the "golden" outputs provided. Input and output image matrics can be visualized in MATLAB, Octave, or similar environments. Instructions can be found in `README.txt`.

Performance Assessment

The 2D convolution kernel nominally involves floating point arithmetic and thus GFLOPS/W is one viable performance metric. The unit of work for relative

comparisons is the generation of a single output pixel. Each input data set includes $M \times N$ such units of work.

1.2 Kernel 2 – Discrete Wavelet Transform (DWT)

1.2.1 Motivation

A wavelet is a single instance of a wave, isolated in time and frequency. A wavelet transform represents a signal as a linear combination of wavelets. The discrete wavelet transform is similar to the Fourier transform, except that it captures both frequency and temporal information. The discrete wavelet transform can be used as a preconditioner to data compression in areas of image and video processing.

The discrete wavelet transform to be implemented is the reversible 5/3 transform used in the JPEG2000 specification for lossless compression [1, 2]. When applied in two dimensions to an image, the discrete wavelet transform is a combination of 1D transforms. The DWT is applied first to each row, then to each column. This may be implemented using a transpose of rows to columns in between each 1D transform.

1.2.2 Description

The input image uses 32-bit unsigned pixels with a dynamic range of 16 bits per pixel. There are three sizes of input images: small (640x480), medium, (1920x1080), and large (3840x2160). The image width is M pixels and the height is N pixels. The 2D discrete wavelet transform is calculated by applying a 1D DWT to the rows and then again to the columns. This may be accomplished by transposing the image between transforms. This process is repeated for each of 30 input frames.

The reversible 5/3 transform takes an input $f[n]$ and produces a low-frequency component $l[n]$ and a high frequency component $h[n]$.

$$h[n] = f[2n+1] - \lfloor \frac{1}{2}(f[2n] + f[2n+2]) \rfloor \tag{1.5}$$

$$l[n] = f[2n] + \lfloor \frac{1}{4}(h[n] + h[n-1]) \rfloor \tag{1.6}$$

The algorithm can be implemented in-place in the following manner. At the end of the computation, the low-frequency component is stored in the even locations and the high-frequency component is stored in the odd locations. Note that the high frequency component must be calculated prior to the low frequency component when computing in-place.

$$f[2n+1] = f[2n+1] - \lfloor \frac{1}{2}(f[2n] + f[2n+2]) \rfloor \tag{1.7}$$

$$f[2n] = f[2n] + \lfloor \frac{1}{4}(f[2n-1] + f[2n+1]) \rfloor \tag{1.8}$$

1.2.3 Specification

Input Data: 30 frames; 16 bits per pixel; small (640x480), medium (1920x1080), large (3840x2160)

Output Data: 30 frames; 16 bits per pixel

Key Operations: Compute a 5/3 discrete wavelet transform of the input image

1.2.4 Assessment

Correctness Assessment

The discrete wavelet transform kernel should be compared with the "golden" outputs provided. Input and output image matrics can be visualized in MAT-LAB, Octave, or similar environments. Instructions can be found in `README.txt`.

Performance Assessment

The unit of work for relative comparisons is the generation of a single output pixel. Each input data set includes $M \times N$ such units of work.

1.3 Kernel 3 – Histogram Equalization

1.3.1 Motivation

The histogram of an image represents the probability distribution function of the pixel values. Histogram equalization increases the contrast of an image by spreading the most frequent intensity values over a larger range.

1.3.2 Description

The input image uses 32-bit unsigned pixels with a dynamic range of 16 bits per pixel. There are three sizes of input images: small (640x480), medium, (1920x1080), and large (3840x2160).

The first step computes a histogram of pixel values of the input image. The histogram places the value of each pixel $f[x, y]$ into one of L uniformly-spaced buckets $h[i]$ (where $L = 2^{16}$). The image width is M pixels and the height is N pixels.

$$h[i] = \sum_{x=1}^{N} \sum_{y=1}^{M} \begin{cases} 1, & \text{if } f[x, y] = i. \\ 0, & \text{otherwise.} \end{cases} \tag{1.9}$$

Next, calculate the cumulative probability distribution function $cdf[j]$ on the histogram $h[i]$.

$$cdf[j] = \sum_{i=1}^{j} h[i] \tag{1.10}$$

Using the cumulative probability distribution function, scale each pixel in the input image to produce an output image $g[x, y]$. Output pixels are 32-bit unsigned integers and $L = 2^{16}$. This process is repeated for each of 30 input frames.

$$g[x, y] = \frac{cdf[f[x, y]] - cdf_{min}}{(N \cdot M) - cdf_{min}} \cdot (L - 1) \tag{1.11}$$

cdf_{min} is the smallest non-zero value of the cumulative probability distribution function.

1.3.3   Specification

Input Data: 30 frames; 16 bits per pixel; small (640x480), medium (1920x1080), large (3840x2160)

Output Data: 30 frames; 16 bits per pixel

Key Operations: Compute the histogram of input pixel values, compute the cumulative distribution of the histogram, and scale the input pixels to the output according to the cumulative distribution such that contrast is maximized.

1.3.4   Assessment

Correctness Assessment

The histogram equalization kernel should be compared with the "golden" outputs provided. Input and output image matrics can be visualized in MATLAB, Octave, or similar environments. Instructions can be found in `README.txt`.

Performance Assessment

The histogram equalization kernel nominally involves floating point arithmetic and thus GFLOPS/W is one viable performance metric. The unit of work for relative comparisons is the generation of a single output pixel. Each input data set includes $M \times N$ such units of work.

Chapter 2

Space-Time Adaptive Processing (STAP)

Radar systems on an airborne platform must often mitigate the impact of ground clutter (i.e., radar returns from the ground or structures), which can overwhelm other signals of interest. Airborne movement further complicates the mitigation process because radar returns are spread in the Doppler dimension. Space-time adaptive processing (STAP) algorithms address clutter by adaptively computing and applying filters in both the spatial (angular) and temporal (Doppler) domains.

The PERFECT Suite's STAP application domain is based upon the extended factored algorithm (EFA) introduced by DiPietro [3]. The Suite's implementations are similar to RT_STAP, a real-time STAP benchmark developed by the MITRE corporation [4]. In particular, the Suite uses the third-order Doppler-factored STAP benchmark case from RT_STAP (i.e., the 'hard' case). An important difference between the Suite and RT_STAP is that the Suite excludes RT_STAP's additional pre-processing.

2.1   Motivation and Background

Space-time adaptive processing largely utilizes standard linear algebra functions, including inner and outer products and linear system solves, as exemplified by the STAP kernel selection. However, there are a few practical considerations that can substantially impact performance. For example, the inner and outer products operate on vectors (space-Doppler snapshots) that are extracted from a larger data set and thus memory access patterns are a key consideration for those kernels. In particular, explicitly forming a large set of contiguous space-Doppler snapshot vectors and then calling inner or outer product functions on that set of vectors may not be the best option. Furthermore, the linear system solves are applied to a large set of relatively small linear systems rather than to a small number of large linear systems. Finally, two of the three kernels operate

on a three-dimensional array and thus some memory reorganization (e.g., corner turns) may be required to obtain high performance.

2.1.1 Data Cube

The input to STAP is a three-dimensional radar data cube consisting of L channels (or phase centers), P pulses, and N samples per pulse. The N samples per pulse are commonly termed range cells or range bins because they sample at range (or time) intervals. Because of the time scales involved, the range dimension is referred to as fast-time and the pulse dimension is referred to as slow-time. The P pulses and L channels add temporal and spatial diversity, respectively, to the acquired data and thus those dimensions correspond to space and time in space-time adaptive processing. We represent the data cube as $\mathbf{x}$ and an individual entry with channel l, range cell n, and pulse p as $\mathbf{x}(l, n, p)$. After the Doppler processing described in Section 2.1.4, the data cube is represented by $\mathbf{X}$ and the entry corresponding to channel l, range cell n, and Doppler bin k is denoted as $\mathbf{X}(l, n, k)$.

2.1.2 Space-Doppler Snapshots

The inner and outer product kernels are applied to space-Doppler snapshot vectors extracted from the Doppler-processed data cube, $\mathbf{X}$. A space-Doppler snapshot vector for range cell n and Doppler bin k includes all channels and the neighboring Doppler bins. For the third-order Doppler-factored case, a neighborhood of three Doppler bins is used and we denote this case by $T_{DOF} = 3$ where T_{DOF} indicates the number of temporal degrees of freedom. Using MATLAB-like notation, the snapshot column vector is given by `vec(X(:,n,k-1:k+1))` where `vec` is a function that vectorizes an array (i.e., it simply returns `a(:)` for input array $\mathbf{a}$). In cases where the Doppler indices would be out-of-bounds, they are wrapped (i.e., -1 wraps to $K - 1$ and K wraps to 0, assuming 0-based indices). The ordering of entries within the snapshot vector is arbitrary, but must be consistent throughout the processing. The snapshot operator is denoted as $\mathbf{S}(n, k)$ such that $\mathbf{S}(n, k)$ can be interpreted as a column vector.

2.1.3 Steering Vectors

STAP will ultimately generate values for the range cell and Doppler bin corresponding to each of D specified steering vectors. The definition of the steering vectors depend upon the geometry of the antenna. From the perspective of the STAP kernels and application, the steering vectors are simply given as inputs and are thus not discussed further. The d-th steering vector has dimension $(L \cdot T_{DOF}) \times 1$ and will be denoted by $\mathbf{v}(d)$.

2.1.4 Doppler Processing (FFTs)

The first stage of EFA converts from the pulse (time) domain to the Doppler spectrum domain by applying a discrete Fourier transform (DFT) of length K to the P pulses for each range cell and channel pair, potentially after having applied a window function to the P pulses. Because the DFT will presumably be implemented via the FFT and the FFT is already a kernel included with the PERFECT Suite, we do not include a separate Doppler processing kernel for the STAP application. However, for completeness we highlight that the incoming data cube may not be stored in pulse-major order (in fact, that is unlikely because pulse is the slowest temporal dimension), so either the FFT would need to be applied in a strided fashion to non-contiguous data or the data cube would need to be transformed into pulse-major order. This step is one of several involving radar data cubes where a corner turn operation may be needed for efficiency.

2.1.5 STAP

The goal of STAP is ultimately to apply an adaptive weighting vector, $\mathbf{w}$, to the snapshot vectors in order to generate an output value for a specific range cell, Doppler bin, and steering direction. Therefore, a single complex output value y will be given by

$$y = \mathbf{w}^H \mathbf{v} \tag{2.1}$$

where $\mathbf{v}$ is the steering direction and the superscript H denotes the Hermitian transpose operator. The optimal weighting vector $\mathbf{w}_{opt}$ is given by

$$\mathbf{w}_{opt} = \gamma \mathbf{R}^{-1} \mathbf{v} \tag{2.2}$$

where $\mathbf{R}$ is the covariance matrix of the interference (including clutter, noise, etc.) and γ is a scalar chosen to provide certain properties, such as constant false alarm rate (CFAR) detection [5].

In practice, the exact covariance matrices are unknown and must be estimated from the data. Specifically, the covariance matrices are estimated by averaging the outer products of a set of training space-Doppler snapshot vectors. Therefore, the covariance estimation comprises the outer product STAP kernel. After computing covariance estimates $\hat{\mathbf{R}}$, the associated adaptive weighting vectors will be computed by solving the linear systems given by $\hat{\mathbf{R}}\hat{\mathbf{w}} = \mathbf{v}$ using the linear system solver kernel. Finally, the adaptive weighting vectors are applied to the snapshot vectors and appropriately scaled during the adaptive weighting kernel, which corresponds to the inner product computations.

2.1.6 Parameter Sets

The total workload and relative contribution of each kernel to the total workload both depend upon the particular parameter set chosen for the data cube and

Table 2.1: STAP kernel parameters for the small, medium, and large test cases.

Parameter	Variable	Small	Medium	Large
Spatial channels/elements	L	4	6	8
Pulses per CPI	P	128	128	128
Doppler bins	K	256	256	256
Range bins	N	512	1024	4096
Range bins per training block	N_R	32	64	64
Temporal degrees of freedom	T_{DOF}	3	3	3
Steering vectors	D	16	16	16

related processing. The PERFECT suite includes small, medium, and large parameter values and associated input and output data sets. The parameters for the three cases are given in Table 2.1.

2.2 STAP Kernel 1 – Outer Product: Covariance Estimation

2.2.1 Description

STAP requires an estimate of the covariance for a given range cell and Doppler bin in order to calculate the associated adaptive weights. One option for making such an estimate is to train based on the acquired data itself. The RT_STAP benchmark employs a block training strategy that corresponds to the covariance estimation kernel. In particular, the range dimension is blocked into M disjoint blocks with N_R contiguous range cells such that $M = N/N_R$. For simplicity, we assume that N_R evenly divides N.

The covariance estimate for range block b and Doppler bin k is then

$$\hat{\mathbf{R}}(b,k) = \frac{1}{N_R} \sum_{r=b \cdot N_R}^{(b+1) \cdot N_R - 1} \mathbf{S}(r,k) \mathbf{S}^H(r,k). \tag{2.3}$$

Thus, the covariance estimation kernel generates $M \times K$ covariance matrices. Note that many training strategies are available. For example, a sliding window could be utilized such that a unique covariance matrix is generated for each range cell n. However, the sliding window strategy would generate $N \times K$ covariance matrices, which would correspondingly increase the computational load for the linear system solver by a factor of N_R. Therefore, the block training strategy is in part an optimization to reduce the computational burden of solving for the adaptive weights. While this benchmark does not focus on training strategies, the RMB rule, named after the authors who observed it, indicates that approximately $2N_R L$ training samples corresponds to a roughly 3 dB performance loss under certain statistical assumptions [6].

12

2.2.2 Specification

Input Data: A 3D complex data cube of dimension $N \times K \times L$.

Output Data: $(N/N_R) \times K$ complex matrices, each of dimension $(L \cdot T_{DOF}) \times (L \cdot T_{DOF})$.

Key Operations: Each output matrix is the mean of N_R outer products of $(L \cdot T_{DOF})$-length vectors. The output matrices are conjugate symmetric, so symmetry can be exploited to reduce computation and/or storage.

2.2.3 Assessment

Correctness Assessment

The outer product kernel can be assessed using the SNR metric described in Section 5.1. For the reference implementation, a single SNR value is computed over all covariance matrices (i.e., over all of the produced output). Although no specific correctness threshold is yet defined, SNR values in the range of 100 dB are reasonable.

Performance Assessment

The outer product kernel nominally involves floating point arithmetic and thus GFLOPS/W is one viable performance metric. The unit of work for relative comparisons is the generation of a single complex covariance matrix of size $(L \cdot T_{DOF}) \times (L \cdot T_{DOF})$, although symmetry may be exploited for more compact storage. Each input data set includes $(N/N_R) \times K$ such units of work.

2.3 STAP Kernel 2 – Linear System Solver: Weight Generation

2.3.1 Description

The covariance matrices $\hat{\mathbf{R}}(b, k)$ generated by the kernel described in Section 2.2 are conjugate symmetric and positive semi-definite. Furthermore, due to receiver noise, the covariance matrices are typically positive definite with a suitable amount of training [7, Chapter 10]. For the purposes of the PERFECT Suite, the covariance matrices can be assumed to be positive definite and conjugate symmetric.

While the RT_STAP benchmark utilizes QR decomposition, any suitable method for solving the linear systems given by

$$\hat{\mathbf{R}}(b, k)\hat{\mathbf{w}}(b, k, d) = \mathbf{v}(d) \tag{2.4}$$

is sufficient. The explicit inverses, $\hat{\mathbf{R}}^{-1}(b, k)$, are not needed and the systems have D associated right-hand sides corresponding to the D steering directions.

Furthermore, an additional weighting scalar $\hat{\gamma}$ can be included to provide desired qualities. In this case, the scalar is given as

$$\hat{\gamma} = (\hat{\mathbf{w}}^H \mathbf{v})^{-1} = ((\hat{\mathbf{R}}^{-1}\mathbf{v})^H \mathbf{v})^{-1} = (\mathbf{v}^H \hat{\mathbf{R}}^{-H} \mathbf{v})^{-1} \tag{2.5}$$

where we have dropped the indices for clarity. $\hat{\mathbf{R}}$ is Hermitian positive definite, so the Hermitian of its inverse, $\hat{\mathbf{R}}^{-H}$, is also positive definite. Therefore, $\mathbf{v}^H \hat{\mathbf{R}}^{-H} \mathbf{v}$ is a real scalar and its inverse is simply $1/(\mathbf{v}^H \hat{\mathbf{R}}^{-H} \mathbf{v})$. With this definition for $\hat{\gamma}$, the weighted results are normalized such that $\hat{\gamma}\hat{\mathbf{w}}^H \mathbf{v} = 1$. Because the weighting scalars are computed from $\hat{\mathbf{w}}$ and $\mathbf{v}$, they can either be computed and stored explicitly or computed on-the-fly as part of the adaptive weighting described in Section 2.4.

2.3.2 Specification

Input Data: $(N/N_R) \times K$ complex matrices, each of dimension $(L \cdot T_{DOF}) \times (L \cdot T_{DOF})$, and D complex steering vectors of length $L \cdot T_{DOF}$.

Output Data: $(N/N_R) \times K \times D$ complex weighting vectors, each of length $L \cdot T_{DOF}$.

Key Operations: Solve $(N/N_R) \times K \times D$ linear systems $\mathbf{Ax} = \mathbf{b}$ where matrix $\mathbf{A}$ is $(L \cdot T_{DOF}) \times (L \cdot T_{DOF})$ and $\mathbf{x}$ and $\mathbf{b}$ are both $(L \cdot T_{DOF})$-length column vectors. The matrices can be assumed to be Hermitian positive definite. A single matrix $\mathbf{A}$ will be reused with D right-hand sides.

2.3.3 Assessment

Correctness Assessment

The system solver kernel can be assessed using the SNR metric described in Section 5.1. For the reference implementation, a single SNR value is computed over all weighting vectors (i.e., over all of the produced output). Although no specific correctness threshold is yet defined, SNR values in the range of 100 dB are reasonable.

Performance Assessment

The system solver kernel nominally involves floating point arithmetic and thus GFLOPS/W is one viable performance metric. The unit of work for relative comparisons is the generation of D complex steering vectors of length $L \cdot T_{DOF}$. Each input data set includes $(N/N_R) \times K$ such units of work.

2.4 STAP Kernel 3 – Inner Product: Adaptive Weighting

2.4.1 Description

Finally, the adaptive weights are applied to the snapshot vectors to form the final STAP output. This step corresponds to a complex inner product and is given by

$$\mathbf{y}(n, k, d) = \hat{\gamma}(b, k, d)\hat{\mathbf{w}}^H(b, k, d)\mathbf{S}(n, k) \qquad (2.6)$$

for range cell n, Doppler bin k, steering direction d, and range block b corresponding to range cell n. The same weighting vector and associated scalar is used for all range cells within a range block and thus the block training strategy also potentially reduces memory traffic during the adaptive weighting, although it does not necessarily impact the number of floating point operations.

2.4.2 Specification

Input Data: A 3D complex data cube of dimension $N \times K \times L$ and $(N/N_R) \times K \times D$ complex weighting vectors, each of length $L \cdot T_{DOF}$.

Output Data: A 3D complex data cube of dimension $N \times K \times D$.

Key Operations: Compute $N \times K \times D$ complex inner products over $L \cdot T_{DOF}$-length vectors. Furthermore, an additional $K \times (N/N_R) \times D$ complex inner products over vectors of length $L \cdot T_{DOF}$ will be required for the scaling given by $\hat{\gamma}$ in Eq. (2.6).

2.4.3 Assessment

Correctness Assessment

The inner product kernel can be assessed using the SNR metric described in Section 5.1. For the reference implementation, a single SNR value is computed over all weighting vectors (i.e., over all of the produced output). Although no specific correctness threshold is yet defined, SNR values in the range of 100 dB are reasonable.

Performance Assessment

The inner product kernel nominally involves floating point arithmetic and thus GFLOPS/W is one viable performance metric. The unit of work for relative comparisons is the generation of a single complex output sample. Each input data set includes $N \times K \times D$ such units of work.

Chapter 3

Synthetic Aperture Radar (SAR)

3.1 Motivation and Background

Synthetic aperture radar (SAR) is a radar-based imaging modality capable of producing high-resolution imagery from an airborne platform. Rather than utilizing a large aperture to achieve high resolution, SAR synthesizes a large aperture using platform motion and then forms an image using data corresponding to the pulses acquired over the synthetic aperture. The amount of data needed for image formation is a function of the radar system parameters, desired image resolution, and coverage area.

The SAR kernels correspond to two alternative methods of image formation. In particular, the two interpolation kernels relate to the polar format algorithm (PFA), which is a Fourier-based approach, and the backprojection kernel represents the inner loop for the backprojection algorithm. Backprojection has a higher computational complexity than PFA, but PFA has some restrictions in terms of imaging geometry compared to the more general backprojection algorithm. For example, PFA assumes that the spherical wavefronts incident upon a scene are planar and the extent to which that assumption holds depends upon the specifics of the collection scenario.

3.1.1 SAR Data

Although SAR can utilize a three-dimensional radar data cube similar to that presented for STAP in Section 2.1.1, we consider only the two-dimensional case of P radar pulses and N complex-valued samples per pulse. In the case of backprojection, the N samples per pulse will be upsampled to N_{BP} samples per pulse prior to the backprojection kernel. The output of SAR processing will be a complex-valued image of dimension $N_x \times N_y$, although in the case of PFA an additional 2D IFFT would need to be performed after the interpolations in

order to produce an output image. The backprojection kernel, on the other hand, will directly produce an output image.

3.1.2 Polar Format Algorithm (PFA)

Data collected during a SAR acquisition corresponds to the Fourier transform of scene reflectivity and thus an image can be reconstructed via inverse Fourier transforms. However, the sampled data points correspond to a polar grid in Fourier space and thus the fast Fourier transform (FFT), which operates on a rectangular grid, cannot be used directly. In order to leverage the FFT, the data is first interpolated onto a rectangular grid. While the required interpolation is notionally two-dimensional, it is typical to assume separability and perform the interpolation as two one-dimensional interpolation steps. The two PERFECT suite SAR interpolation kernels correspond to these one-dimensional interpolations that achieve the polar to rectangular coordinate conversion. Both PFA interpolation kernels utilize a truncated sinc interpolation footprint of width T_{PFA}. If we assume that the number of pulses, samples per pulse, and pixels-per-side of the formed image are all approximately N, then the computational complexity of PFA is given by $\mathcal{O}(N^2 \log N)$.

3.1.3 Backprojection (BP)

The backprojection, or backpropagation, algorithm is a method of image formation that directly integrates a contribution from each pulse into each pixel. Thus, the computational complexity for BP is $\mathcal{O}(P \cdot N_x \cdot N_y)$. For the nominal case where the number of pulses and pixels-per-side of the formed image are comparable (e.g., N), the computational complexity of backprojection is $\mathcal{O}(N^3)$.

Although it is not included as a separate kernel, it is common to upsample data in the range dimension prior to backprojection in order to effectively improve the interpolation quality (i.e., high order interpolation during upsampling followed by lower-order interpolation during backprojection). The upsampling can be performed via an FFT with zero-padding in the Fourier domain and is thus already covered by the 1D FFT in the required kernel section. The backprojection kernel operates on data that has already been upsampled and thus the samples-per-pulse value will be $N_{BP} \geq N$.

3.1.4 Parameter Sets

The total workload of each kernel depends upon the particular parameter set chosen for the input data set and related processing. The PERFECT suite includes small, medium, and large parameter values and associated input and output data sets. The parameters for the three cases are given in Table 3.1.

17

Table 3.1: SAR kernel parameters.

Parameter	Variable	Small	Medium	Large
Pulses	P	512	1024	2048
Range bins	N	512	1024	2048
Upsampled range bins (BP)	N_{BP}	4096	8192	16384
Resampled range bins (PFA)	N_{PFA}	512	1024	2048
Resampled pulses (PFA)	P_{PFA}	512	1024	2048
Image size	$N_x \times N_y$	512×512	1024×1024	2048×2048
Carrier frequency	f_c	10 GHz	10 GHz	10 GHz
PFA interpolation width	T_{PFA}	13	13	13

3.2 SAR Kernel 1 – Interpolation 1: PFA Range Interpolation

3.2.1 Description

Both PFA kernels utilize the normalized sinc function given by

$$\text{sinc}(x) = \frac{\sin(\pi x)}{\pi x}$$

to interpolate from a given input coordinate grid to a given output coordinate grid. The first interpolation is applied along the range dimension and the second is applied along the azimuth (pulse) dimension. Furthermore, the interpolation kernel is weighted by a real-valued window function $\mathbf{w}$. The interpolation for both SAR kernels takes the following form [7, Chapter 6]:

$$\mathbf{g}(m) = \frac{\Delta y}{\Delta x} \sum_{k=-(T_{PFA}-1)/2}^{(T_{PFA}-1)/2} \mathbf{f}(k + n(\mathbf{y_m}))\mathbf{w}(k)\,\text{sinc}\left(\frac{\mathbf{y}_m - \mathbf{x}_{k+n(\mathbf{y}_m)}}{\Delta x}\right) \quad (3.1)$$

where $\mathbf{f}$ and $\mathbf{g}$ are the input and output functions, $\mathbf{x}$ and $\mathbf{y}$ are the input and output coordinates, Δx and Δy are the average spacing for the input and output coordinates, $\mathbf{w}$ is the window function defined over the interpolation kernel, and $n(\mathbf{y}_m)$ is the index of the coordinate in $\mathbf{x}$ nearest to $\mathbf{y}_m$. In other words, we resample the function $\mathbf{f}$ defined at coordinates $\mathbf{x}$ to the function $\mathbf{g}$ defined at coordinates $\mathbf{y}$. In the case where the interpolation kernel extends outside the domain of $\mathbf{f}$ (e.g., when $k + n(\mathbf{y_m}) < 0$), values of zero replace $\mathbf{f}$.

For PFA range interpolation, Eq. (3.1) is applied along the range (fast-time) dimension. The input sample coordinates vary with pulse, but they are uniformly spaced for a given pulse. Furthermore, it can be assumed that the input coordinates are monotonically increasing. The equidistant input sampling can be exploited during interpolation when computing the nearest input neighbor index $n(\mathbf{y}_m)$ to output coordinate $\mathbf{y}_m$. The reference implementation exploits

the uniform sample spacing of the input coordinates to compute $n()$ via a closed form expression.

3.2.2 Specification

Input Data: A 2D $P \times N$ complex data matrix, a 2D $P \times N$ array of input coordinates, a 1D array of N_{PFA} output coordinates, and a PFA interpolation width (T_{PFA}).

Output Data: A resampled 2D $P \times N_{PFA}$ complex data matrix.

Key Operations: For each complex output sample, compute the nearest neighbor among the input coordinates and apply a T_{PFA}-width truncated sinc interpolation kernel to the associated complex input data values. The input coordinates can be assumed to be both monotonically increasing and uniformly spaced for a given pulse, but may vary from pulse to pulse.

3.2.3 Assessment

Correctness Assessment

The PFA range interpolation kernel can be assessed using the SNR metric described in Section 5.1. For the reference implementation, a single SNR value is computed over all samples (i.e., over all of the produced output). Although no specific correctness threshold is defined, SNR values in the range of 100 dB are reasonable.

Performance Assessment

The PFA range interpolation kernel nominally involves floating point arithmetic and thus GFLOPS/W is one viable performance metric. The unit of work for relative comparisons is the generation of a single complex output sample. Each input data set includes $P \times N_{PFA}$ such units of work.

3.3 SAR Kernel 2 – Interpolation 2: PFA Azimuth Interpolation

3.3.1 Description

PFA azimuth interpolation is similar to range interpolation and still utilizes Eq. (3.1) for the interpolation model. The azimuth interpolation is applied to the output of the range interpolation kernel and resamples the data along the azimuth (pulse) dimension. However, for azimuth interpolation, it is no longer the case that the input spacing is generally uniform for a given range

frequency.[1] Therefore, the computation of the nearest input neighbor index for a given output coordinate may be more general for azimuth interpolation than range interpolation. The reference implementation employs a binary search method to compute $n()$ for the azimuth case, but any method of computing $n()$ over a non-uniform input sample grid is acceptable. It can be assumed that the input coordinates are monotonically increasing.

3.3.2 Specification

Input Data: A 2D $P \times N_{PFA}$ complex data matrix, a 2D $P \times N_{PFA}$ array of input coordinates, a 1D array of P_{PFA} output coordinates, and a PFA interpolation width (T_{PFA}).

Output Data: A resampled 2D $P_{PFA} \times N_{PFA}$ complex data matrix.

Key Operations: For each complex output sample, compute the nearest neighbor among the input coordinates and apply a T_{PFA}-width truncated sinc interpolation kernel to the complex input data values. The input coordinates can be assumed to be monotonically increasing, but not necessarily uniformly spaced, for a given range frequency. Furthermore, the input coordinates may vary from range frequency to range frequency.

3.3.3 Assessment

Correctness Assessment

The PFA azimuth interpolation kernel can be assessed using the SNR metric described in Section 5.1. For the reference implementation, a single SNR value is computed over all samples (i.e., over all of the produced output). Although no specific correctness threshold is defined, SNR values in the range of 100 dB are reasonable.

Furthermore, the output of the PFA azimuth interpolation can be visualized as an image after applying a 2D inverse FFT. The TAV suite includes the script `generate_pfa_image.m` for applying the 2D IFFT and displaying the resulting image. This script can be run in MATLAB, Octave, or other similar environments. Sample output from the reference kernel implementation is shown in Figure 3.1. This data set included three simulated point reflectors that manifest as two-dimensional sinc functions in the output image.

Performance Assessment

The PFA azimuth interpolation kernel nominally involves floating point arithmetic and thus GFLOPS/W is one viable performance metric. The unit of work

---

[1]The initial input data set does have uniform sample spacing along the input azimuth coordinates, but that is due to having used an idealized simulation to generate the data. Platform velocity or trajectory variation during an acquisition would generate non uniformly spaced azimuth samples.
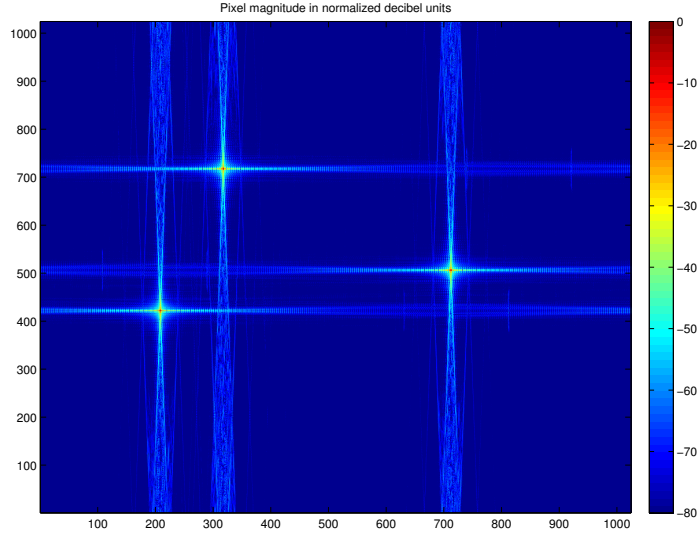
Figure 3.1: Sample output image generated by applying a 2D inverse FFT to the PFA kernel 2 (azimuth interpolation) output. Pixel values correspond to normalized magnitude in decibel scale.

for relative comparisons is the generation of a single complex output sample. Each input data set includes $P_{PFA} \times N_{PFA}$ such units of work.

3.4 SAR Kernel 3 – Backprojection

3.4.1 Description

The backprojection algorithm is described in many papers and textbooks [8, 9, 10], often in the context of similar image formation algorithms from medical imaging via x-ray computed tomography. Backprojection takes as input the location of the imaging platform for each pulse, the location of each output pixel, and the $P \times N_{BP}$ data set. For pixel k and pulse p, backprojection consists of the following steps:

- Compute the distance from the platform to the pixel
- Convert the distance to an associated position (range) in the data set
- Sample at the computed range via linear interpolation
- Scale the sampled value by a matched filter to form the pixel contribution
- Accumulate the contribution into the pixel.

Given input data $\mathbf{x}$ of dimension $P \times N_{BP}$, per-pulse platform positions $\mathbf{v}$, and output image $\mathbf{y}$, the kth output pixel with location $\mathbf{a}_k = (x_k, y_k, z_k)$ is given by

$$\mathbf{y}_k = \sum_{p=1}^{P} \tilde{\mathbf{x}}(p, r(p,k)) \cdot e^{j \cdot 2 \cdot k_u \cdot r(p,k)} \tag{3.2}$$

where $r(p,k) = ||\mathbf{a}_k - \mathbf{v}_p||$ is the distance from the platform location at pulse p to pixel k and the tilde on $\mathbf{x}$ indicates linear interpolation in range to estimate the value at position $r(p,k)$. The value k_u is equal to $2\pi f_c/c$ where f_c is the carrier frequency of the radar waveform and c is the speed of light. The complex exponential $e^{j\theta}$ is equivalent to $\cos(\theta) + j\sin(\theta)$ via Euler's formula and thus a sine and cosine computation is implied in Equation (3.2) for each pixel-pulse pair. While k_u is a constant from the perspective of an implementation, its value will impact the magnitude of the argument for the sine and cosine calculations and thus may have a direct impact on performance due to argument reduction.

The backprojection algorithm is shown in pseudocode form in Algorithm 1 where the linear interpolation mechanics are made explicit. The values R_0 and ΔR in Algorithm 1 denote the distance to the first range bin and the distance per range bin, respectively. R_0 and ΔR are constants from the perspective of the implementation and are used to convert the computed distance (range) into an array index for linear interpolation.

Algorithm 1 Backprojection pseudocode.

1: **for all** pixels k **do**
2:     $\mathbf{y}_k = 0$
3:     **for all** pulses p **do**
4:         $R = ||\mathbf{a}_k - \mathbf{v}_p||$ /* Distance from platform to voxel */
5:         $\text{bin} = \lfloor (R - R_0)/\Delta R \rfloor$ /* Range bin (integer) */
6:         **if** $\text{bin} \in [0, N_{BP} - 2]$ **then**
7:             $w = (R - R_0)/\Delta R - \text{bin}$
            /* Data sampled using linear interpolation */
8:             $s = (1-w) \cdot \mathbf{x}(p, \text{bin}) + w \cdot \mathbf{x}(p, \text{bin}+1)$
9:             $\mathbf{y}_k = \mathbf{y}_k + s \cdot e^{j \cdot k_u \cdot R}$
10:         **end if**
11:     **end for**
12: **end for**

3.4.2   Specification

Input Data: A 2D $P \times N_{BP}$ complex data matrix, a vector of P three-dimensional platform positions, and output pixel coordinates. While backprojection can generate pixel data for arbitrary coordinates (given proper data support), the output image is centered at the origin and is assumed rectangular with uniform pixel spacing. The output image is parameterized by the pixel
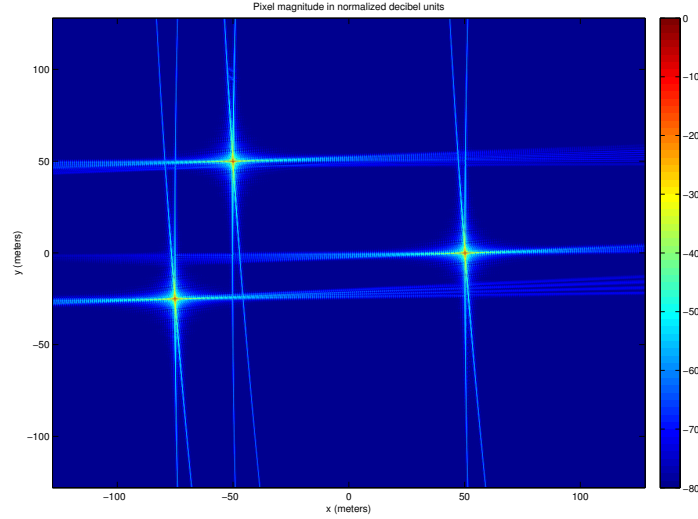
Figure 3.2: Sample output image generated by the SAR backprojection kernel. Pixel values correspond to normalized magnitude in decibel scale.

spacing Δx and Δy for the x and y dimensions, respectively.

Output Data: A complex image of dimension $N_x \times N_y$.

Key Operations: For each pixel and pulse, there is a distance calculation (including a square root), a linear interpolation, a complex exponential computation, and a complex multiplication and accumulation.

3.4.3 Assessment

Correctness Assessment

The backprojection kernel can be assessed using the SNR metric described in Section 5.1. For the reference implementation, a single SNR value is computed over all pixels (i.e., over all of the produced output). Although no specific correctness threshold is defined, SNR values in the range of 100 dB are reasonable.

Furthermore, the backprojection kernel generates an output image and that image can be visualized and compared against the "golden" output image. The TAV suite includes the script `display_bp_image.m` for displaying the generated images. This script can be run in MATLAB, Octave, or other similar environments. Sample output from the reference kernel implementation is shown in Figure 3.2. Note that the output images generated by PFA and BP differ in several respects and thus should not be compared against one another for validation purposes.

Performance Assessment

The backprojection kernel nominally involves floating point arithmetic and thus GFLOPS/W is one viable performance metric. The unit of work for relative comparisons is the application of a single backprojection (i.e., accumulating the contribution of a single pulse into a single pixel). Each input data set includes $P \times N_x \times N_y$ such units of work.

Chapter 4

Required Kernels

4.1 1D FFT

4.1.1 Motivation

The one-dimensional fast Fourier transform (FFT) is a fast method of computing the discrete Fourier transform (DFT) that is pervasive in signal processing applications.

4.1.2 Description

The k-th element of the N-point DFT is given as

$$X_k = \sum_{n=0}^{N-1} x_n e^{-j2\pi nk/N}, \quad k = 0, \ldots, N-1 \tag{4.1}$$

where $j = \sqrt{-1}$ and the e term is the complex exponential. This formulation assumes that the input and output sequence lengths are identical, although in practice satisfying that condition may involve zero-padding the the input sequence.

The inverse DFT is then

$$x_n = \frac{1}{N} \sum_{k=0}^{N-1} X_k e^{j2\pi nk/N}, \quad n = 0, \ldots, N-1. \tag{4.2}$$

In this formulation, normalization is included as the $1/N$ term in the inverse DFT, although normalization can be handled in several ways.

4.1.3 Specification

For the purposes of the PERFECT Suite, it can be assumed that N is a power of two and that the x_n input samples are complex-valued. Values of N for the

Small, Medium, and Large input test cases are 256, 4096, and 32768 respectively. Due to the similarity in the DFT and inverse DFT equations, considering only the forward DFT is sufficient. While the FFT is a widely used method for accelerating the computation of the DFT, any valid method of computing the DFT is acceptable.

4.1.4 Assessment

Correctness Assessment

The 1D FFT kernel can be assessed using the SNR metric described in Section 5.1. Although no specific correctness threshold is yet defined, SNR values in the range of 100 dB are reasonable.

Performance Assessment

The 1D FFT kernel nominally involves floating point arithmetic and thus GFLOPS/W is one viable performance metric. The unit of work for relative comparisons is the generation of a single complex 1D FFT of length N. The FFT input data is generated randomly and thus the number of units of work can be adjusted as needed.

4.2 2D FFT

4.2.1 Motivation

Whereas the one-dimensional DFT is pervasive in signal processing, the two-dimensional DFT is widely used in image processing and image formation, such as in the polar format algorithm used with synthetic aperture radar.

4.2.2 Description

The two-dimensional DFT is given by

$$X_{k,l} = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} x_{m,n} e^{-j2\pi\left(\frac{km}{M} + \frac{ln}{N}\right)} \tag{4.3}$$

for $k = 0, \ldots, M-1$, $l = 0, \ldots, N-1$, $j = \sqrt{-1}$, and the e term is the complex exponential. This formulation assumes that the input and output dimensions are identical, although in practice satisfying that condition may involve zero-padding the the input sequence.

Similarly, the two-dimensional inverse DFT is given by

$$X_{m,n} = \frac{1}{MN} \sum_{k=0}^{M-1} \sum_{l=0}^{N-1} X_{k,l} e^{j2\pi\left(\frac{km}{M} + \frac{ln}{N}\right)} \tag{4.4}$$

for $m = 0, \ldots, M-1$ and $n = 0, \ldots, N-1$.

4.2.3 Specification

For the purposes of the PERFECT Suite, it can be assumed that M and N are powers of two and that the input samples $x_{m,n}$ are complex-valued. Values of MxN for the Small, Medium, and Large input test cases are 1024x1024, 4096x4096, and 8192x8192 respectively. Furthermore, due to the similarity in the DFT and inverse DFT equations, considering only the forward DFT is sufficient. While the FFT is a widely used method for accelerating the computation of the DFT, any valid method of computing the DFT is acceptable.

4.2.4 Assessment

Correctness Assessment

The 2D FFT kernel can be assessed using the SNR metric described in Section 5.1. Although no specific correctness threshold is yet defined, SNR values in the range of 100 dB are reasonable.

Performance Assessment

The 2D FFT kernel nominally involves floating point arithmetic and thus GFLOPS/W is one viable performance metric. The unit of work for relative comparisons is the generation of a single complex 2D FFT of size $M \times N$. The FFT input data is generated randomly and thus the number of units of work can be adjusted as needed.

4.3 Sorting

4.3.1 Motivation

While sorting is not as pervasive as the FFT for signal processing, it does occur in some contexts, such as computing ordered statistics for ordered statistic constant false alarm rate (OS-CFAR) detectors [11].

4.3.2 Description

In an embedded context, operations are often applied on relatively small windows of data rather than on very large data sets and thus the sorting kernel is focused on applying many relatively small sorting operations rather than a single large sorting operation.

Additionally, it is often required to "carry" additional auxiliary data or metadata throughout the sort operation such that the metadata remains associated with the original key.

4.3.3 Specification

For the purposes of this sorting kernel, the original indices of the keys will form the metadata such that the original location of a given key is available after sorting. Retaining the key metadata would then enable reversing the sorting operation after applying intermediate processing or rearranging a separate array consistently with the first. The length of the input data for the Small, Medium, and Large test cases are 9, 64, and 1024 elements respectively. A batch of 1024 input arrays should be sorted, either sequentially or in parallel.

4.3.4 Assessment

Correctness Assessment

From the perspective of correctness, the sorting kernel is an example where exact reproducibility is expected, up to ambiguities in the original values. Because stable sorting is not required, two identical keys may have a swapped order after sorting in two conforming implementations, which would then correspond to the metadata also being stored in a swapped order.

Performance Assessment

The sorting kernel does not involve floating point arithmetic, but does involve floating point comparisons. Thus, considering floating point comparisons to be floating point operations, GFLOPS/W is a viable performance metric. The unit of work for relative comparisons is the sorting of a single array. The sorting input data is generated randomly and thus the number of units of work can be adjusted as needed.

Chapter 5

Correctness and Assessment Metrics

For many of the kernels and applications, identically reproducing the results generated by the reference implementations is unreasonable and unnecessary. Firstly, the non-commutativity of floating point arithmetic and many other issues imply that bitwise equivalence of two independent implementations is unlikely. Secondly, it is the goal of the TAV team to allow and encourage innovation on the part of the performers, which may involve developing and exploiting approximations or other suitable optimizations. As a result, some criteria are needed in order to constrain the implementations in a way that both provides flexibility to the performers as well as produces results that would still be acceptable for the associated application.

   In this section, we present criteria that can be used to evaluate performer implementations and yield feedback on the quality of the results produced by the implementation. There are no strict boundaries defined to separate acceptable and unacceptable results, but the value of the metric can form an additional axis along which to evaluate an implementation. The correctness criteria will evolve as more kernels are defined and based on performer feedback.

5.1   Signal-to-Noise Ratio (SNR)

One useful metric is a signal-to-noise ratio (SNR) where we treat as noise differences in produced results relative to some reference value. SNR is decibel-scaled and is given

$$SNR_{dB} = 10 \log_{10} \left(\frac{\sum_{k=1}^{N} |r_k|^2}{\sum_{k=1}^{N} |r_k - t_k|^2} \right) \tag{5.1}$$

where r_k and t_k are the reference and test values for the k-th element, respectively, and N is the number of entries being compared. The values r_k and t_k

can be complex, in which case the $|\cdot|$ operator corresponds to the magnitude of the complex number. Large, positive values for SNR represent close agreement between the reference and test values. Each 20 dB corresponds roughly to a single digit of accuracy between the values, so 80 dB represents about four digits of agreement. Exact agreement between arrays yields a division by zero, which can be treated as an SNR of positive infinity. For the reference STAP kernels, the SNR relative to data produced with a separate, double-precision implementation is over 100 dB.

One limitation of SNR is that it is not very sensitive to large errors in a small number of values when N is large. Thus, some important errors can be masked by the SNR calculation. This effect can be partially addressed by computing SNR over smaller windows of data, which produces multiple, localized metrics. The assessment portion of each kernel description will specify the preferred method of using the SNR, if applicable.

Bibliography

[1] ISO/IEC 15444-1: Information technology–JPEG 2000 image coding system: Core coding system, 2004.

[2] Michael D. Adams and Rabab Ward. Wavelet transforms in the JPEG-2000 standard. In *In Proc. of IEEE Pacific Rim Conference on Communications, Computers and Signal Processing*, pages 160–163, 2001.

[3] R. C. DiPietro. Extended factored space-time processing for airborne radar systems. In *Signals, Systems and Computers, 1992. 1992 Conference Record of The Twenty-Sixth Asilomar Conference on*, volume 1, pages 425–430, 1992.

[4] K. C. Cain, J. A. Torres, and R. T. Williams. RT_STAP: Real-time space-time adaptive processing benchmark. Technical Report RL-TR-97-173, The MITRE Corporation, October 1997.

[5] W.L. Melvin. A STAP overview. *Aerospace and Electronic Systems Magazine, IEEE*, 19(1):19–35, 2004.

[6] I. S. Reed, J. D. Mallett, and L. E. Brennan. Rapid convergence rate in adaptive arrays. *IEEE Transactions on Aerospace and Electronic Systems*, AES-10(6):853–863, 1974.

[7] William L. Melvin and James A. Scheer, editors. *Principles of Modern Radar: Advanced Techniques*. SciTech Publishing, 2013.

[8] L. A. Gorham and L. J. Moore. SAR image formation toolbox for MAT-LAB. In *Proceedings of SPIE, Algorithms for Synthetic Aperture Radar Imagery XVII*, volume 7699, 2010.

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

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

[11] H. Rohling. Radar CFAR thresholding in clutter and multiple target situations. *IEEE Transactions on Aerospace and Electronic Systems*, AES-19(4):608–621, 1983.