# PERFECT Benchmark Suite Manual

Kevin Barker (PNNL)
Thomas Benson (GTRI)
Dan Campbell (GTRI)
David Ediger (GTRI)
Nitin Gawande (PNNL)
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 1.0.1

# Contents


**1 Introduction** **4**

**2 PERFECT Application 1 (PA1)** **6**
2.1 Kernel 1 – 2D Convolution . . . . . 6
2.1.1 Motivation . . . . . 6
2.1.2 Description . . . . . 6
2.1.3 Specification . . . . . 7
2.1.4 Assessment . . . . . 7
2.2 Kernel 2 – Discrete Wavelet Transform (DWT) . . . . . 8
2.2.1 Motivation . . . . . 8
2.2.2 Description . . . . . 8
2.2.3 Specification . . . . . 9
2.2.4 Assessment . . . . . 9
2.3 Kernel 3 – Histogram Equalization . . . . . 9
2.3.1 Motivation . . . . . 9
2.3.2 Description . . . . . 9
2.3.3 Specification . . . . . 10
2.3.4 Assessment . . . . . 10
2.4 PA1 Application . . . . . 10

**3 Space-Time Adaptive Processing (STAP)** **11**
3.1 Motivation and Background . . . . . 11
3.1.1 Data Cube . . . . . 12
3.1.2 Space-Doppler Snapshots . . . . . 12
3.1.3 Steering Vectors . . . . . 12
3.1.4 Doppler Processing (FFTs) . . . . . 13
3.1.5 STAP . . . . . 13
3.1.6 Parameter Sets . . . . . 13
3.2 STAP Kernel 1 – Outer Product:
Covariance Estimation . . . . . 14
3.2.1 Description . . . . . 14
3.2.2 Specification . . . . . 15
3.2.3 Assessment . . . . . 15

3.3 STAP Kernel 2 – Linear System Solver: Weight Generation . . . 15
3.3.1 Description . . . 15
3.3.2 Specification . . . 16
3.3.3 Assessment . . . 16
3.4 STAP Kernel 3 – Inner Product: Adaptive Weighting . . . 17
3.4.1 Description . . . 17
3.4.2 Specification . . . 17
3.4.3 Assessment . . . 17
3.5 STAP Application . . . 18
3.5.1 Description . . . 18
3.5.2 Specification . . . 18
3.5.3 Assessment . . . 18

**4 Synthetic Aperture Radar (SAR) 20**
4.1 Motivation and Background . . . 20
4.1.1 SAR Data . . . 20
4.1.2 Polar Format Algorithm (PFA) . . . 21
4.1.3 Backprojection (BP) . . . 21
4.1.4 Parameter Sets . . . 21
4.2 SAR Kernel 1 – Interpolation 1: PFA Range Interpolation . . . 22
4.2.1 Description . . . 22
4.2.2 Specification . . . 23
4.2.3 Assessment . . . 23
4.3 SAR Kernel 2 – Interpolation 2: PFA Azimuth Interpolation . . . 23
4.3.1 Description . . . 23
4.3.2 Specification . . . 24
4.3.3 Assessment . . . 24
4.4 SAR Kernel 3 – Backprojection . . . 25
4.4.1 Description . . . 25
4.4.2 Specification . . . 26
4.4.3 Assessment . . . 27

**5 Wide Area Motion Imagery (WAMI) 29**
5.1 Motivation and Background . . . 29
5.1.1 Parameter Sets . . . 29
5.2 WAMI Kernel 1 – Debayer . . . 29
5.2.1 Description . . . 29
5.2.2 Specification . . . 30
5.2.3 Assessment . . . 32
5.2.4 Correctness Assessment . . . 32
5.2.5 Performance Assessment . . . 32

5.3 WAMI Kernel 2 – Image Registration
Lucas-Kanade Algorithm . . . 32
5.3.1 Description . . . 32
5.3.2 Specification . . . 33
5.3.3 Assessment . . . 33
5.4 WAMI Kernel 3 – Change Detection
Gaussian Mixture Models . . . 34
5.4.1 Description . . . 34
5.4.2 Specification . . . 35
5.4.3 Assessment . . . 37
5.5 WAMI Application . . . 37
5.5.1 Description . . . 37
5.5.2 Specification . . . 38
5.5.3 Assessment . . . 38

**6 Required Kernels** **40**
6.1 1D FFT . . . 40
6.1.1 Motivation . . . 40
6.1.2 Description . . . 40
6.1.3 Specification . . . 40
6.1.4 Assessment . . . 41
6.2 2D FFT . . . 41
6.2.1 Motivation . . . 41
6.2.2 Description . . . 41
6.2.3 Specification . . . 42
6.2.4 Assessment . . . 42
6.3 Sorting . . . 42
6.3.1 Motivation . . . 42
6.3.2 Description . . . 42
6.3.3 Specification . . . 43
6.3.4 Assessment . . . 43

**7 Correctness and Assessment Metrics** **44**
7.1 Signal-to-Noise Ratio (SNR) . . . 44

# Chapter 1

# Introduction

The PERFECT Suite [1] is a collection of applications and kernels representing domains of interest to the PERFECT program.

As shown in Figure 1.1, the Suite is divided into five major domains. Each domain contains three kernels that represent segments of computational importance for a representative application on the domain. The first four areas represent specific application domains. They are the so-called Perfect Application 1, an Image processing application which has kernels that work on massive data sets; Space Time Adaptive Processing; Synthetic Aperture Radar; and Wide Area Motion Imaging. The final "Required Kernels" contains kernels (Fast Fourier Transforms and sorting algorithms) that are important for multiple application domains.

Each kernel was selected based on the following considerations. A kernel must be computationally important; it should provide reasonable coverage of the domain it represents; the kernel must be algorithmically interesting; and it should appear in only one application domain.

| Application Domains | Kernels |
|---|---|
| PERFECT Application 1 | Discrete Wavelet Transform |
| | 2 D Convolutions |
| | Histogram Equalization |
| Space Time Adaptive Processing | System Solver |
| | Inner Product |
| | Outer Product |
| Synthetic Aperture Radar | Interpolation 1 |
| | Interpolation 2 |
| | Back Projection (Non-Fourier SAR) |
| Wide Area Motion Imaging | Debayer |
| | Image Registration |
| | Change Detection |
| Required Kernels | Sort |
| | FFT 1D |
| | FFT 2D |

Figure 1.1: Summary of the Perfect Benchmark Suite.

# Chapter 2

# PERFECT Application 1 (PA1)

## 2.1 Kernel 1 – 2D Convolution

### 2.1.1 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.

### 2.1.2 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 $h[x]$.

$$h[x] = \begin{bmatrix} 1 & 3 & 4 & 5 & 6 & 5 & 4 & 3 & 1 \end{bmatrix} \tag{2.1}$$

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

$$g[x, y] = h' \cdot h \tag{2.2}$$

$$g[x, y] = \begin{bmatrix} 1 & 3 & 4 & 5 & 6 & 5 & 4 & 3 & 1 \\ 3 & 9 & 12 & 15 & 18 & 15 & 12 & 9 & 3 \\ 4 & 12 & 16 & 20 & 24 & 20 & 16 & 12 & 4 \\ 5 & 15 & 20 & 25 & 30 & 25 & 20 & 15 & 5 \\ 6 & 18 & 24 & 30 & 36 & 30 & 24 & 18 & 6 \\ 5 & 15 & 20 & 25 & 30 & 25 & 20 & 15 & 5 \\ 4 & 12 & 16 & 20 & 24 & 20 & 16 & 12 & 4 \\ 3 & 9 & 12 & 15 & 18 & 15 & 12 & 9 & 3 \\ 1 & 3 & 4 & 5 & 6 & 5 & 4 & 3 & 1 \end{bmatrix} \tag{2.3}$$

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.

$$f[x, y] * g[x, y] = \frac{1}{1024} \sum_{n_1=-4}^{4} \sum_{n_2=-4}^{4} f[n_1, n_2] \cdot g[x - n_1, y - n_2] \tag{2.4}$$

### 2.1.3 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. 2.3 applied

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

### 2.1.4 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.

## 2.2 Kernel 2 – Discrete Wavelet Transform (DWT)

### 2.2.1 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 [2, 3]. 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.

### 2.2.2 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 $h[n]$.

$$h[n] = f[2n+1] - \lfloor \frac{1}{2}(f[2n] + f[2n+2]) \rfloor \tag{2.5}$$

$$l[n] = f[2n] + \lfloor \frac{1}{4}(h[n] + h[n-1]) \rfloor \tag{2.6}$$

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.

$$f[2n+1] = f[2n+1] - \lfloor \frac{1}{2}(f[2n] + f[2n+2]) \rfloor \tag{2.7}$$

$$f[2n] = f[2n] + \lfloor \frac{1}{4}(f[2n-1] + f[2n+1]) \rfloor \tag{2.8}$$

### 2.2.3 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

### 2.2.4 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 MATLAB, 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.

## 2.3 Kernel 3 – Histogram Equalization

### 2.3.1 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.

### 2.3.2 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.

$$h[i] = \sum_{x=1}^{N} \sum_{y=1}^{M} \begin{cases} 1, & \text{if } f[x, y] = i. \\ 0, & \text{otherwise.} \end{cases} \tag{2.9}$$

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

$$cdf[j] = \sum_{i=1}^{j} h[i] \tag{2.10}$$

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.

$$g[x, y] = \frac{cdf[f[x, y]] - cdf_{min}}{(N \cdot M) - cdf_{min}} \cdot (L - 1) \tag{2.11}$$

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

### 2.3.3 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.

### 2.3.4 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.

## 2.4 PA1 Application

# Chapter 3

# 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 [4]. The Suite's implementations are similar to RT_STAP, a real-time STAP benchmark developed by the MITRE corporation [5]. 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.

## 3.1 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.

### 3.1.1 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 3.1.4, 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 $\mathbf{X}(l, n, k)$.

### 3.1.2 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 `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.

### 3.1.3 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 $\mathbf{v}(d)$.

### 3.1.4 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.

### 3.1.5 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

$$y = \mathbf{w}^H \mathbf{v} \tag{3.1}$$

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

$$\mathbf{w}_{opt} = \gamma \mathbf{R}^{-1} \mathbf{v} \tag{3.2}$$

where $\mathbf{R}$ is the covariance matrix of the interference (including clutter, noise, etc.) and $\gamma$ is a scalar chosen to provide certain properties, such as constant false alarm rate (CFAR) detection [6].

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.

### 3.1.6 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 3.1: 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 3.1.

## 3.2 STAP Kernel 1 – Outer Product: Covariance Estimation

### 3.2.1 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

$$\hat{\mathbf{R}}(b,k) = \frac{1}{N_R} \sum_{r=b\cdot N_R}^{(b+1)\cdot N_R - 1} \mathbf{S}(r,k)\mathbf{S}^H(r,k). \tag{3.3}$$

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 $2N_RL$ training samples corresponds to a roughly 3 dB performance loss under certain statistical assumptions [7].

### 3.2.2 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.

### 3.2.3 Assessment

#### Correctness Assessment

The outer product kernel can be assessed using the SNR metric described in Section 7.1. 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.

## 3.3 STAP Kernel 2 – Linear System Solver: Weight Generation

### 3.3.1 Description

The covariance matrices $\hat{\mathbf{R}}(b,k)$ generated by the kernel described in Section 3.2 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 [8, 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

$$\hat{\mathbf{R}}(b,k)\hat{\mathbf{w}}(b,k,d) = \mathbf{v}(d) \tag{3.4}$$

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

$$\hat{\gamma} = (\hat{\mathbf{w}}^H \mathbf{v})^{-1} = ((\hat{\mathbf{R}}^{-1}\mathbf{v})^H \mathbf{v})^{-1} = (\mathbf{v}^H \hat{\mathbf{R}}^{-H} \mathbf{v})^{-1} \tag{3.5}$$

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 3.4.

### 3.3.2 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.

### 3.3.3 Assessment

#### Correctness Assessment

The system solver kernel can be assessed using the SNR metric described in Section 7.1. 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.

## 3.4 STAP Kernel 3 – Inner Product: Adaptive Weighting

### 3.4.1 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

$$\mathbf{y}(n,k,d) = \hat{\gamma}(b,k,d)\hat{\mathbf{w}}^H(b,k,d)\mathbf{S}(n,k) \tag{3.6}$$

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.

### 3.4.2 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. (3.6).

### 3.4.3 Assessment

#### Correctness Assessment

The inner product kernel can be assessed using the SNR metric described in Section 7.1. 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.

## 3.5 STAP Application

The kernels and operations described above can be merged into a full space-time adaptive processing pipeline, or STAP application.

### 3.5.1 Description

In addition to the three PERFECT suite STAP kernels, the STAP application adds the Doppler processing step described in Section 3.1.4. The application arranges the operations in the following order, with the outputs from one stage forming the input to the next:

1. Doppler processing – Corner turns, windowing, and FFTs
2. Kernel 1 – Outer products – Covariance estimation
3. Kernel 2 – Linear system solves – Weight generation
4. Kernel 3 – Inner product – Adaptive weighting

The corner turns included in the Doppler processing stage convert the data to pulse-major order to apply FFTs to contiguous array and then re-convert the data to range-major order prior to kernel 1 so that the kernel 1 implementation remains the same for the kernel benchmark and application. However, the corner turns are applied only for convenience and an optimized implementation could arrange the data cube differently and modify the kernel implementations accordingly.

### 3.5.2 Specification

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

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

**Key Operations:** The Doppler processing stage involves corner turns, windowing (i.e., element-wise multiplications), and FFTs. The three kernels following Doppler processing involve the same operations as described in Sections 3.2.2, 3.3.2, and 3.4.2.

### 3.5.3 Assessment

#### Correctness Assessment

The full application can be assessed using the SNR metric described in Section 7.1. For the reference implementation, a single SNR value is computed over all output values. The SNR threshold for the full application could resaonably be lower than for individual kernels due to the cumulative loss of precision from one kernel to the next. For the reference implementation, the SNR relative

to a fully double precision implementation is 80 dB, which is reasonable relative to a threshold of 100 dB for the individual kernels.

**Performance Assessment**

The STAP application 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. Note, however, that workload scales with other parameters, such as $T_{DOF}$. Therefore, the units of work are only comparable between similarly parameterized variants of STAP.

# Chapter 4

# Synthetic Aperture Radar (SAR)

## 4.1 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.

### 4.1.1 SAR Data

Although SAR can utilize a three-dimensional radar data cube similar to that presented for STAP in Section 3.1.1, 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.

### 4.1.2 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 $\mathcal{O}(N^2 \log N)$.

### 4.1.3 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 $N_{BP} \geq N$.

### 4.1.4 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 4.1.

Table 4.1: 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 | $N_x \times N_y$ | $512 \times 512$ | $1024 \times 1024$ | $2048 \times 2048$ |
| Carrier frequency | $f_c$ | 10 GHz | 10 GHz | 10 GHz |
| PFA interpolation width | $T_{PFA}$ | 13 | 13 | 13 |

## 4.2 SAR Kernel 1 – Interpolation 1: PFA Range Interpolation

### 4.2.1 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 [8, Chapter 6]:

$$\mathbf{g}(m) = \frac{\Delta y}{\Delta x} \sum_{k=-(T_{PFA}-1)/2}^{(T_{PFA}-1)/2} \mathbf{f}(k + n(\mathbf{y_m}))\mathbf{w}(k)\,\text{sinc}\left(\frac{\mathbf{y}_m - \mathbf{x}_{k+n(\mathbf{y}_m)}}{\Delta x}\right) \quad (4.1)$$

where $\mathbf{f}$ and $\mathbf{g}$ are the input and output functions, $\mathbf{x}$ and $\mathbf{y}$ are the input and output coordinates, $\Delta x$ and $\Delta y$ 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. (4.1) 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.

### 4.2.2 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.

### 4.2.3 Assessment

#### Correctness Assessment

The PFA range interpolation kernel can be assessed using the SNR metric described in Section 7.1. 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.

## 4.3 SAR Kernel 2 – Interpolation 2: PFA Azimuth Interpolation

### 4.3.1 Description

PFA azimuth interpolation is similar to range interpolation and still utilizes Eq. (4.1) 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.

### 4.3.2 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.

### 4.3.3 Assessment

#### Correctness Assessment

The PFA azimuth interpolation kernel can be assessed using the SNR metric described in Section 7.1. 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 4.1. 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.

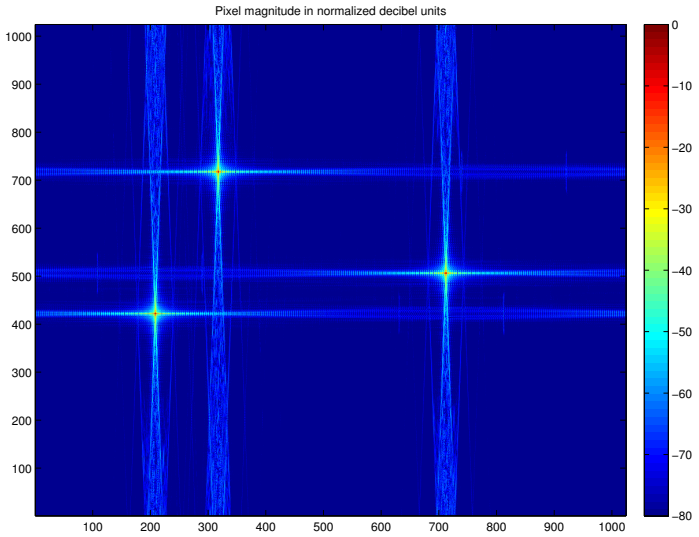


Figure 4.1: Sample output image generated by applying a 2D inverse FFT to the PFA kernel 2 (azimuth interpolation) output. Pixel values correspond to normalized magnitude in decibel scale.

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.

## 4.4 SAR Kernel 3 – Backprojection

### 4.4.1 Description

The backprojection algorithm is described in many papers and textbooks [9, 10, 11], 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 $k$th output pixel with location $\mathbf{a}_k = (x_k, y_k, z_k)$ is given by

$$\mathbf{y}_k = \sum_{p=1}^{P} \tilde{\mathbf{x}}(p, r(p,k)) \cdot e^{j \cdot 2 \cdot k_u \cdot r(p,k)} \tag{4.2}$$

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 (4.2) 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 $\Delta R$ in Algorithm 1 denote the distance to the first range bin and the distance per range bin, respectively. $R_0$ and $\Delta R$ 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.

```
for all pixels k do
    y_k = 0
    for all pulses p do
        R = ||a_k − v_p||  /* Distance from platform to voxel */
        bin = ⌊(R − R_0)/ΔR⌋  /* Range bin (integer) */
        if bin ∈ [0, N_BP − 2] then
            w = (R − R_0)/ΔR − bin
            /* Data sampled using linear interpolation */
            s = (1 − w) · x(p, bin) + w · x(p, bin + 1)
            y_k = y_k + s · e^{j·k_u·R}
        end if
    end for
end for
```

### 4.4.2 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

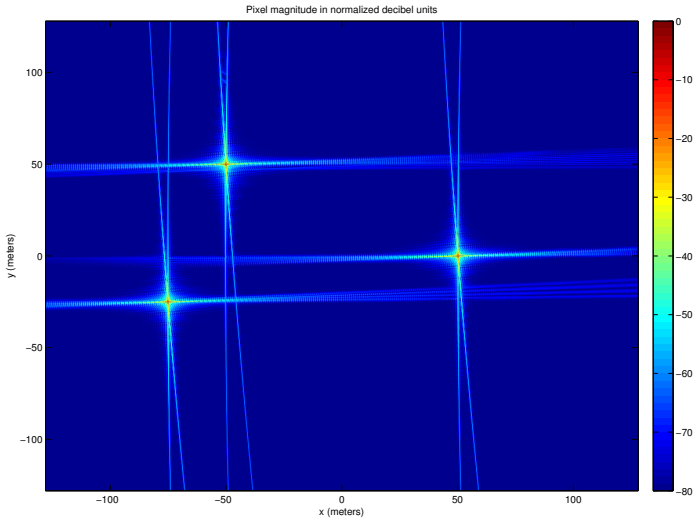


Figure 4.2: Sample output image generated by the SAR backprojection kernel. Pixel values correspond to normalized magnitude in decibel scale.

spacing $\Delta x$ and $\Delta y$ 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.

### 4.4.3 Assessment

#### Correctness Assessment

The backprojection kernel can be assessed using the SNR metric described in Section 7.1. 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 4.2. 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 5

# Wide Area Motion Imagery (WAMI)

## 5.1 Motivation and Background

Modern imaging sensors produce a very large amount of data from which useful information must be extracted. For example, the DARPA-funded ARGUS-IS system acquires a 1.8 gigapixel image at a rate of greater than 12 Hz [12]. The wide area motion imagery (WAMI) kernels in the PERFECT suite correspond to three typical processing steps of a WAMI processing chain, including RGB image generation via the debayer algorithm, image registration via the Lucas-Kanade algorithm, and change detection (or background subtraction) via a Gaussian mixture model based algorithm.

### 5.1.1 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 related processing. The PERFECT suite includes small, medium, and large parameter values and associated input and output data sets. The parameters for the three WAMI cases are given in Table 5.1.

## 5.2 WAMI Kernel 1 – Debayer

### 5.2.1 Description

Many digital cameras utilizing a single charge coupled device (CCD) array include a color array filter such that each CCD element samples only a single color (red, green, or blue). The colors not sampled at a given pixel are then estimated via interpolation in order to yield red, green, and blue (RGB) samples for each pixel. One common color array filter, the Bayer array, is shown in Table 5.2.

Table 5.1: WAMI kernel parameters for the small, medium, and large test cases.

| Parameter | Variable | Small | Medium | Large |
|---|---|---|---|---|
| Image height (pixels) | $Y$ | 512 | 1024 | 2048 |
| Image width (pixels) | $X$ | 512 | 1024 | 2048 |
| Number of frames (Kernel 3) | $N$ | 5 | 5 | 5 |

Table 5.2: Bayer color array filter pattern. R, G, and B represent red, green, and blue.

| | | | | |
|---|---|---|---|---|
| R | G | R | G | $\cdots$ |
| G | B | G | B | $\cdots$ |
| R | G | R | G | $\cdots$ |
| G | B | G | B | $\cdots$ |
| $\cdots$ | $\cdots$ | $\cdots$ | $\cdots$ | $\cdots$ |

The debayer kernel takes as input a Bayer array image with one sample per pixel and returns an image with RGB samples per pixel where the missing colors have been estimated via interpolation using the approach described in [13]. The interpolation mask utilizes a $5 \times 5$ footprint centered on the pixel for which a value is being estimated. Thus, the output sample for row $r$ and column $c$ is given by

$$\mathbf{d}(r,c) = \texttt{clamp}\left(\left\lfloor \sum_{i=0}^{4}\sum_{j=0}^{4} \mathbf{h}(i,j)\mathbf{b}(r-2+i, c-2+j) \right\rfloor, 0, \texttt{MAX}\right) \tag{5.1}$$

where $\mathbf{h}$ is the appropriate interpolation kernel, $\mathbf{b}$ is the original Bayer image, $\mathbf{d}$ is the debayer image color sample associated with $\mathbf{h}$, and `MAX` is the maximum integral value for a pixel. `MAX` is defined subsequently in the specification section. The clamp operation returns 0 if a negative value is produced by the interpolation and `MAX` if a value exceeding `MAX` is produced. For colors sampled directly in the Bayer image, no interpolation is needed. The interpolation weights for the other cases are given in Table 5.3. For simplicity, the PERFECT suite only considers pixels for which the full interpolation footprint can be applied and thus the output image has two fewer pixels along each edge (i.e., the output image has four fewer pixels in each dimension).

### 5.2.2 Specification

The input and output images will consist of unsigned 16-bit integer pixel samples and thus `MAX` is defined as $2^{16} - 1$. However, the incoming pixels can be assumed to have a bit depth of only 12 bits, despite being represented as unsigned 16-bit values. The intermediate calculations can be performed using either integer or floating point arithmetic, but the correctness of an output pixel

Table 5.3: Debayer interpolation footprints as a function of the interpolated color and its location. All coefficients should be further divided by 8. R, G, and B indicate red, green, and blue samples. ER, OR, EC, and OC refer to even (E) or odd (O) rows (R) and columns (C). These values have been reproduced from Figure 2 in [13].

| | | | | |
|---|---|---|---|---|
| 0 | 0 | -1 | 0 | 0 |
| 0 | 0 | 2 | 0 | 0 |
| -1 | 2 | 4 | 2 | -1 |
| 0 | 0 | 2 | 0 | 0 |
| 0 | 0 | -1 | 0 | 0 |

G at R locations

| | | | | |
|---|---|---|---|---|
| 0 | 0 | -1 | 0 | 0 |
| 0 | 0 | 2 | 0 | 0 |
| -1 | 2 | 4 | 2 | -1 |
| 0 | 0 | 2 | 0 | 0 |
| 0 | 0 | -1 | 0 | 0 |

G at B locations

| | | | | |
|---|---|---|---|---|
| 0 | 0 | 0.5 | 0 | 0 |
| 0 | -1 | 0 | -1 | 0 |
| -1 | 4 | 5 | 4 | -1 |
| 0 | -1 | 0 | -1 | 0 |
| 0 | 0 | 0.5 | 0 | 0 |

R at G in ER, OC

| | | | | |
|---|---|---|---|---|
| 0 | 0 | -1 | 0 | 0 |
| 0 | -1 | 4 | -1 | 0 |
| 0.5 | 0 | 5 | 0 | 0.5 |
| 0 | -1 | 4 | -1 | 0 |
| 0 | 0 | -1 | 0 | 0 |

R at G in OR, EC

| | | | | |
|---|---|---|---|---|
| 0 | 0 | -1.5 | 0 | 0 |
| 0 | 2 | 0 | 2 | 0 |
| -1.5 | 0 | 6 | 0 | -1.5 |
| 0 | 2 | 0 | 2 | 0 |
| 0 | 0 | -1.5 | 0 | 0 |

R at B locations

| | | | | |
|---|---|---|---|---|
| 0 | 0 | 0.5 | 0 | 0 |
| 0 | -1 | 0 | -1 | 0 |
| -1 | 4 | 5 | 4 | -1 |
| 0 | -1 | 0 | -1 | 0 |
| 0 | 0 | 0.5 | 0 | 0 |

B at G in OR, EC

| | | | | |
|---|---|---|---|---|
| 0 | 0 | -1 | 0 | 0 |
| 0 | -1 | 4 | -1 | 0 |
| 0.5 | 0 | 5 | 0 | 0.5 |
| 0 | -1 | 4 | -1 | 0 |
| 0 | 0 | -1 | 0 | 0 |

B at G in ER, OC

| | | | | |
|---|---|---|---|---|
| 0 | 0 | -1.5 | 0 | 0 |
| 0 | 2 | 0 | 2 | 0 |
| -1.5 | 0 | 6 | 0 | -1.5 |
| 0 | 2 | 0 | 2 | 0 |
| 0 | 0 | -1.5 | 0 | 0 |

B at R locations

value is defined by Equation 5.1, so care must be taken to avoid, for example, underflow, overflow, and truncation prior to accumulation.

**Input Data:** A 2D array (image) of $Y \times X$ pixels sampled in a Bayer pattern.

**Output Data:** A 3D array (image) of dimension $3 \times (Y-4) \times (X-4)$ where each of the $(Y-4) \times (X-4)$ pixels is defined by a triple of red, green, and blue (RGB) values.

**Key Operations:** For each of $(Y-4) \times (X-4)$ pixels, the two colors that were not natively sampled will be interpolated via a $5 \times 5$ interpolation footprint, which requires both multiplication and summation. For points in the interpolation footprint that are defined to be zero, no operations need to be performed.

### 5.2.3 Assessment

### 5.2.4 Correctness Assessment

Because of the limited precision of the incoming pixels values, the specific weights used for the debayer kernel, and the use of unsigned 16-bit integers for output, the output pixel values should match the "golden" output values exactly.

### 5.2.5 Performance Assessment

The debayer kernel may involve either integer-only or floating point arithmetic, depending upon the implementation. Thus, either GFLOPS/W for a floating point implementatin or GOPS/W for an integer implementation are viable performance metrics. The unit of work for relative comparisons is the number of output pixels. Each input data set includes $(Y-4) \times (X-4)$ such units of work.

## 5.3 WAMI Kernel 2 – Image Registration Lucas-Kanade Algorithm

### 5.3.1 Description

Image alignment is the process of moving and warping a template image so as to minimize the difference between the template and the original image. Image alignment can be used to track regions of interest in motion across multiple frames of video, as well as stabilizing a series of images containing jitter. The customary approach to alignment is gradient descent in which the warp parameters are iteratively improved.

The image alignment kernel will implement several steps of the Lucas-Kanade [14] algorithm as detailed in [15]. The required steps include computing a) the warped gradients (Step 3 in the paper), b) the steepest descent images (Step 5 in the paper), and c) the Hessian matrix (Step 6 in the paper).

$\mathbf{W(x;\ p)}$ is the parameterized set of allowed warps, where $\mathbf{p} = (p_1, p_2, ...p_n)^T$ is a vector of parameters. We will consider the set of affine warps:

$$\mathbf{W(x;\ p)} = \begin{pmatrix} (1+p_1)\cdot x & + & p_3 \cdot y & + & p_5 \\ p_2 \cdot x & + & (1+p_4)\cdot y & + & p_6 \end{pmatrix} \tag{5.2}$$

with 6 parameters $\mathbf{p} = (p_1, p_2, p_3, p_4, p_5, p_6)^T$ [16] that are defined below. Equation 5.2 has the Jacobian:

$$\frac{\partial \mathbf{W}}{\partial \mathbf{p}} = \begin{pmatrix} x & 0 & y & 0 & 1 & 0 \\ 0 & x & 0 & y & 0 & 1 \end{pmatrix}. \tag{5.3}$$

The Hessian matrix $H$ is the $n \times n$ matrix:

$$H = \sum_{\mathbf{x}} \left[ \nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \right]^T \left[ \nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \right] \tag{5.4}$$

where $\nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}}$ is referred to as the steepest descent images (matrix). It is the product of the gradient and the Jacobian. For the Jacobian given in Equation 5.3, the Hessian matrix $H$ will be $6 \times 6$.

MATLAB code corresponding to the Baker and Matthews paper [15] is available at `http://www.ri.cmu.edu/research_project_detail.html?project_id=515`.

### 5.3.2 Specification

Input images will consist single precision floating point values. The input images are horizontal and vertical gradients of a source image, therefore pixel values will be both negative and positive. Input warp parameters will be real-valued and are given in the coordinate system of the input image. The Jacobian chosen depends only on pixel coordinates (Eqn. 5.3). The pixel coordinate system of the input image is $x$ from 0 to $X-1$ and $y$ from 0 to $Y-1$. The output is the $6 \times 6$ Hessian matrix consisting of real numbers.

The warp parameters for implementations will be:

$$\mathbf{p} = (-0.035, 0.01, -0.01, -0.035, 5.5, 5.5)^T$$

and the region of interest (template) is the full frame ($Y \times X$). When warping the gradients, bilinear interpolation is suggested. If one or more of the four neighbor pixels is outside of the image boundary, choose the interpolated value to be zero.

**Input Data:** 2D arrays (images) of $Y \times X$ pixels containing the X- and Y-gradients of the sample image and warp parameters $\mathbf{p}$; small (512x512), medium (1024x1024), large (2048x2048)

**Output Data:** $6 \times 6$ Hessian matrix.

**Key Operations:** Warp both X- and Y-gradients according to Equation 5.2 and $\mathbf{p}$. Compute steepest descent images from warped gradients and Jacobian. Compute Hessian matrix according to Equation 5.4.

### 5.3.3 Assessment

#### Correctness Assessment

The image alignment kernel should be compared against the "golden" output Hessian matrix and can be assessed using the SNR metric described in Section 7.1. Although no specific correctness threshold is yet defined, SNR values in the range of 100 dB are reasonable.

**Performance Assessment**

The image alignment kernel nominally involves floating point arithmetic and thus GFLOPS/W is one viable performance metric. The unit of work for relative comparisons is the number of pixels in the region of interest. Each input data set includes $Y \times X$ such units of work.

## 5.4 WAMI Kernel 3 – Change Detection Gaussian Mixture Models

### 5.4.1 Description

When processing a time series of images, scene changes from one frame to the next are often of interest. An accurate mask of "meaningfully" changed pixels (i.e., not different only due to noise, lighting changes, etc.) has many uses, including identifying objects for further processing, enabling compression techniques, etc. The change detection kernel for the PERFECT suite is based upon a Gaussian mixture model as described in [17].

The Gaussian mixture model associates $K$ Gaussian distributions, each with an additional weight factor, with each pixel in the image. The Gaussians and their respective weights adapt to each new frame in order to accommodate long-term scene changes. Depending on the weights, a certain proportion of the Gaussians are assumed to be associated with background processes and thus those pixels not determined to be consistent with background processes are labeled as foreground pixels.

Although the Gaussian mixture model (GMM) approach can be applied to color images, the PERFECT suite assumes that only greyscale luminance images will be used. Each pixel will be represented by an unsigned 16-bit integer. The Gaussian mixture model includes state information developed throughout the course of processing frames. In particular, each pixel $j$ has $K$ associated Gaussian distributions with mean, variance, and weight values $\mu_{k,j}$, $\sigma^2_{k,j}$, and $w_{k,j}$, respectively, for the $k$-th distribution. The Gaussian parameters update dynamically from frame-to-frame according to the algorithm depicted via pseudocode in Algorithm 2. GMM depends upon the following parameters, which will all be given as inputs for the PERFECT suite:

- $\alpha$ : Learning rate. Higher values for $\alpha$ correspond to faster adaptivity to background changes at the potential cost of susceptibility to noise and over-fitting.
- $T_\sigma$ : Sigma threshold. A threshold, in standard deviations, below which pixels are determined to match an existing model.
- $\omega_{\texttt{init}}$ : Initial weight used for a newly created model. This value should be relatively low.

Table 5.4: WAMI kernel parameters for the small, medium, and large test cases.

| Parameter | Variable | Value |
|---|---|---|
| Learning rate | $\alpha$ | 0.01 |
| Sigma threshold | $T_\sigma$ | 2.5 |
| Initial model weight | $\omega_{\texttt{init}}$ | 0.01 |
| Initial model variance | $\sigma^2_{\texttt{init}}$ | 6400 |
| Background portion | $T_{\texttt{background}}$ | 0.9 |

- $\sigma^2_{\texttt{init}}$ : Initial variance used for a newly created model. This value should be relatively high.
- $T_{\texttt{background}}$ : Expected portion of the image that should be explained by the background models. Higher values of $T_{\texttt{background}}$ correspond to fewer detected changes and lower values of $T_{\texttt{background}}$ correspond to more detected changes.

The values used for the above parameters are shown in Table 5.4.

Furthermore, $\eta$ as shown in Algorithm 2 is the Gaussian probability density function:

$$\eta(X_j|\mu_{k,j},\sigma_{k,j}) = \frac{1}{\sigma_{k,j}\sqrt{2\pi}} \exp\left(\frac{-(X_j-\mu_{k,j})^2}{2\sigma^2_{k,j}}\right)$$

The pixels classified as foreground are those that are considered to have changed for the purposes of change detection. Although additional processing, such as morphological operations, would likely follow the GMM algorithm in a real application, the PERFECT suite includes only GMM for the change detection kernel.

For the purposes of the PERFECT suite, an initial set of Gaussian parameters will be provided as input to the kernel such that the model has effectively already been trained. Therefore, full initialization of the model is not required.

### 5.4.2 Specification

**Input Data:** Three 3D $Y \times X \times K$ single-precision floating point arrays containing the mean, variance, and weight for each of the $K$ Gaussian models and $N$ $Y \times X$ frames of greylevel pixels stored as 16-bit unsigned integers.

**Output Data:** $N$ foreground masks, each of size $Y \times X$, where each 8-bit unsigned pixel is set to one for foreground (i.e., changed) pixels and zero otherwise. Furthermore, the mean, variance, and weights for the model are updated for each frame to maintain a trained state.

**Key Operations:** For each pixel, either identify the distribution to which it

**Algorithm 2** Pseudocode for a Gaussian mixture model (GMM) based change detection algorithm. The algorithm and notation is modeled after that given in [17]. These steps assume that a model already exists and thus no initialization is required; this is the case for the PERFECT WAMI change detection kernel.

**for all** pixels $X_j$ **do**
  **for all** Gaussians $k$ **do**
    match $\leftarrow 0$
    /* Check for distribution matches */
    **if** match == 0 and $|X_j - \mu_{k,j}|/\sigma_{k,j} < T_\sigma$ **then**
      /* First matching distribution; adjust its parameters. */
      $\rho \leftarrow \alpha\eta(X_j|\mu_{k,j}, \sigma_{k,j})$
      $\mu_{k,j} \leftarrow (1-\rho)\mu_{k,j} + \rho X_j$
      $\sigma^2_{k,j} \leftarrow (1-\rho)\sigma^2_{k,j} + \rho(X_j - \mu_{k,j})^2$
      $\omega_{k,j} \leftarrow (1-\alpha)\omega_{k,j} + \alpha$
      match $\leftarrow 1$
    **else**
      /* Non-matching distribution; decay its weight. */
      $\omega_{k,j} \leftarrow (1-\alpha)\omega_{k,j}$
    **end if**
  **end for**
  /* If no match, then replace the least likely distribution. */
  **if** match == 0 **then**
    $\mu_{K-1,j} \leftarrow X_j$
    $\sigma^2_{K-1,j} \leftarrow \sigma^2_{\texttt{init}}$
    $\omega_{K-1,j} \leftarrow \omega_{\texttt{init}}$
  **end if**
  Renormalize weights such that $\sum_{k=0}^{K-1} \omega_{k,t} = 1$
  Sort Gaussians in descending order by $\omega_{k,t}/\sigma_{k,t}$
  /* Determine the number of background distributions, $B$ */
  $B \leftarrow \texttt{argmin}_b \left(\sum_{k=0}^{b} \omega_{k,t} > T_{\texttt{background}}\right)$
  **if** pixel matched one of first $B$ distributions **then**
    Classify $X_j$ as background
  **else**
    Classify $X_j$ as foreground
  **end if**
**end for**

belongs or create a new distribution, update the model parameters, and classify the pixel as either background or foreground.

### 5.4.3 Assessment

#### Correctness Assessment

Whereas the foreground map generated by GMM is binary from the perspective of a pixel (background or foreground), many floating point operations are involved in the GMM process and thus an exact match against a "golden" data set would not be feasible in general. One option would be to compare the state variables ($\mu$, $\sigma$, $\omega$) using the SNR metric, but instead correctness is assessed directly in the foreground map domain. Because the foreground map would often undergo an erosion or median filter prior to further processing to reduce noise, the PERFECT suite assessment routine first applies an opening operation (erosion followed by dilation) to both the test and "golden" foreground masks and then counts the number of pixels with differing values between the two masks.

#### Performance Assessment

The change detection kernel nominally involves floating point arithmetic and thus GFLOPS/W is one viable performance metric, although this kernel is particularly memory-intensive. The unit of work for relative comparisons is the number of pixels in the image. Each input data set includes $N \times Y \times X$ such units of work.

## 5.5 WAMI Application

The kernels and operations described above can be merged into a full wide area motion imaging processing pipeline, or WAMI application.

### 5.5.1 Description

Merging the previously described kernels into a WAMI application requires adding an RGB to grayscale conversion after the Debayer stage and completing the Lucas-Kanade implementation for image registration relative to the partial implemenation described in Section 5.3. The stages for the full application are arranged as follows:

1. Kernel 1 – Debayer
2. RGB to grayscale conversion
3. Modified Kernel 2 – Lucas-Kanade – Image registation
4. Kernel 3 – Gaussian mixture models – Change detection

The RGB to grayscale conversion is simply a linear combination of the R, G, and B channels to generate a single grayscale output channel as input to the image registration step. The image registration step for the full application implements the steps from [15] that were missing from the kernel described in Section 5.3. In particular, Kernel 2 implemented most of the computationally intensive operations, but the algorithm from [15] involves applying nine steps iteratively. The remainder of the steps are described in the referenced paper and included in the MATLAB implemenation available, as of this writing, at `http://www.ri.cmu.edu/research_project_detail.html?project_id=515`.

### 5.5.2 Specification

Because the initial kernel for the WAMI application is the Debayer stage, the inputs for the Debayer kernel and full application are the same in terms of the image. In addition, state is provided for the image registration and change detection kernels so that they can produce meaningful output on a small number of frames; however, including state is only for convenience and a true deployment of the application would construct state (such as the Gaussian mixture model parameters) over time.

**Input Data:** A 3D array of $N$ images with $Y \times X$ pixels per image sampled in a Bayer pattern.

**Output Data:** $N$ foreground masks, each of size $(Y - 4) \times (X - 4)$, where each 8-bit unsigned pixel is set to one for foreground (i.e., changed) pixels and zero otherwise.

**Key Operations:** The majority of the workload is still captured by the three PERFECT suite kernels and described in Sections 5.2.2, 5.3.2, and 5.4.2. Additionally, the RGB to grayscale conversion involves a per-pixel linear combination of the RGB channels and the image registration stage involves several additional steps relative to those described in Section 5.3.2.

### 5.5.3 Assessment

#### Correctness Assessment

Whereas the foreground map generated by the application is binary from the perspective of a pixel (background or foreground), many floating point operations are involved in the process and thus an exact match against a "golden" data set would not be feasible in general. As a simple metric, the included correctness assessment checked for the percentage of misclassified pixels relative to a golden data set and includes a nominal error threshold of one percent. As discussed in Section 5.4.3, morphological operations would typically be applied to the output foreground mask, so noise-like differences relative to the golden output would generally be acceptable whereas structural differences (e.g., an

entire missing foreground object) would not be acceptable.

**Performance Assessment**

The entire WAMI processing pipeline involves both integer and floating point operations, so OPS/W, where an OP can be either integer or floating point, would be one viable performance metric. The unit of work for relative comparisons is the number of pixels in the post-Debayer image. Each input data set includes $N \times (Y - 4) \times (X - 4)$ such units of work.

# Chapter 6

# Required Kernels

## 6.1 1D FFT

### 6.1.1 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.

### 6.1.2 Description

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

$$X_k = \sum_{n=0}^{N-1} x_n e^{-j2\pi nk/N}, \quad k = 0, \ldots, N-1 \tag{6.1}$$

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

$$x_n = \frac{1}{N} \sum_{k=0}^{N-1} X_k e^{j2\pi nk/N}, \quad n = 0, \ldots, N-1. \tag{6.2}$$

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

### 6.1.3 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.

### 6.1.4 Assessment

#### Correctness Assessment

The 1D FFT kernel can be assessed using the SNR metric described in Section 7.1. 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.

## 6.2 2D FFT

### 6.2.1 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.

### 6.2.2 Description

The two-dimensional DFT is given by

$$X_{k,l} = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} x_{m,n} e^{-j2\pi\left(\frac{km}{M} + \frac{ln}{N}\right)} \tag{6.3}$$

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

$$X_{m,n} = \frac{1}{MN} \sum_{k=0}^{M-1} \sum_{l=0}^{N-1} X_{k,l} e^{j2\pi\left(\frac{km}{M} + \frac{ln}{N}\right)} \tag{6.4}$$

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

### 6.2.3 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 $M$x$N$ 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.

### 6.2.4 Assessment

#### Correctness Assessment

The 2D FFT kernel can be assessed using the SNR metric described in Section 7.1. 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.

## 6.3 Sorting

### 6.3.1 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 [18].

### 6.3.2 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.

### 6.3.3 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.

### 6.3.4 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 7

# 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.

## 7.1 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

$$SNR_{dB} = 10 \log_{10} \left( \frac{\sum_{k=1}^{N} |r_k|^2}{\sum_{k=1}^{N} |r_k - t_k|^2} \right) \tag{7.1}$$

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] Kevin Barker, Thomas Benson, Dan Campbell, David Ediger, Roberto Gioiosa, Adolfy Hoisie, Darren Kerbyson, Joseph Manzano, Andres Marquez, Leon Song, Nathan R. Tallent, and Antonino Tumeo. *PERFECT (Power Efficiency Revolution For Embedded Computing Technologies) Benchmark Suite Manual*. Pacific Northwest National Laboratory and Georgia Tech Research Institute, December 2013. `http://hpc.pnnl.gov/projects/PERFECT/`.

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

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

[4] 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.

[5] 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.

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

[7] 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.

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

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

[10] 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.

[11] 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.

[12] Brian Leininger, Jonathan Edwards, John Antoniades, David Chester, Dan Haas, Eric Liu, Mark Stevens, Charlie Gershfield, Mike Braun, James D. Targove, Steve Wein, Paul Brewer, Donald G. Madden, and Khurram Hassan Shafique. Autonomous real-time ground ubiquitous surveillance-imaging system (ARGUS-IS). volume 6981, pages 69810H–69810H–11, 2008.

[13] Henrique S. Malvar, Li wei He, and Ross Cutler. High-quality linear interpolation for demosaicing of Bayer-patterned color images. In *Proc. of IEEE ICASSP*, volume 3, pages 485–488, 2004.

[14] Bruce D. Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of the 7th international joint conference on Artificial intelligence - Volume 2*, IJCAI'81, pages 674–679, 1981.

[15] Simon Baker and Iain Matthews. Lucas-kanade 20 years on: A unifying framework. *Int. J. Comput. Vision*, 56(3):221–255, February 2004.

[16] James R. Bergen, P. Anandan, Keith J. Hanna, and Rajesh Hingorani. Hierarchical model-based motion estimation. In *Proceedings of the Second European Conference on Computer Vision*, ECCV '92, pages 237–252, 1992.

[17] Chris Stauffer and W.E.L. Grimson. Adaptive background mixture models for real-time tracking. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '99, pages 246–252, 1999.

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