SANDIA REPORT

SAND2005-5232
Unlimited Release
Printed September 2005

IFP V4.0: A Polar-Reformatting Image Formation Processor for Synthetic Aperture Radar

Paul H. Eichel

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia is a multiprogram laboratory operated by Sandia Corporation,
a Lockheed Martin Company, for the United States Department of Energy's
National Nuclear Security Administration under Contract DE-AC04-94-AL85000.

Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.

Available to DOE and DOE contractors from
U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Telephone: (865) 576-8401
Facsimile: (865) 576-5728
E-Mail: reports@adonis.osti.gov
Online ordering: http://www.doe.gov/bridge

Available to the public from
U.S. Department of Commerce
National Technical Information Service
5285 Port Royal Rd
Springfield, VA 22161

Telephone: (800) 553-6847
Facsimile: (703) 605-6900
E-Mail: orders@ntis.fedworld.gov
Online ordering: http://www.ntis.gov/ordering.htm

SAND2005-5232
Unlimited Release
Printed September 2005

IFP V4.0: A Polar-Reformatting Image Formation Processor for Synthetic Aperture Radar

Paul Eichel
Signal Processing & Research Department
Sandia National Laboratories
P.O. Box 5800
Albuquerque, NM 87185-1207
pheiche@sandia.gov

Abstract

IFP V4.0 is the fourth generation of an extraordinarily powerful and flexible image formation processor for spotlight mode synthetic aperture radar. It has been successfully utilized in processing phase histories from numerous radars and has been instrumental in the development of many new capabilities for spotlight mode SAR. This document provides a brief history of the development of IFP, a full exposition of the signal processing steps involved, and a short user's manual for the software implementing this latest iteration.

Acknowledgment

Many individuals at Sandia National Labs have contributed to the development of IFP over the years. We make some effort to chronicle the major contributors in the History section of this report, but just to be unequivocal, IFP would certainly not exist without the efforts of Dennis Ghiglia, Gary Mastin, and Paul Thompson.

Contents

Figures

Preface

It is indeed a welcome addition to the body of knowledge that Paul Eichel has undertaken the challenge of documenting the history of R&D efforts that led to the Image Formation Processor. In addition, he has achieved the lofty goal of documenting the various software enhancements and capabilities, software revisions, signal processing equations, implementations, references, and operator guidance. Also, I thank Paul for giving me the opportunity to write this preface.

Oftentimes the adventures and misadventures, the successes and failures, and the paths taken become obscured over time. It is easy to forget or confuse the intricate entwinement of professional and personal lives encountered over ones career. I can say without hesitation that this was indeed a special time in my life, and I'm sure my colleagues and friends would agree it was a special time for them as well. To those readers not directly involved in this R&D, this report provides a welcome overview of how it came to be and how it is. To those of us that were directly involved during these exciting times (and mostly for whom this preface is written), I'm sure we felt a sense of adventure and discovery with potentially lasting impact not likely to be repeated during the course of our careers. This report should serve as a reminder of that.

I sincerely believe that the reason we did have so much fun and were able to accomplish what we did was due in large part to the freedom we were given by Sandia management and our Department of Energy sponsor. Without their support and encouragement from perspectives of employee empowerment, facilities, and substantial funding, it is doubtful that the team effort would have worked as well as it did. We thank them for that. And speaking of the team, I wish to express my sincere gratitude for the privilege of working with some of the finest people I have ever met. Unfortunately, there are too many to mention individually, but you know who you are! We will always remain friends and colleagues and I wish you continued success and more fun times in the future.

Dennis C. Ghiglia
Sandia National Laboratories
Albuquerque, NM
1978-1997

Vexcel Corporation
Boulder, CO
1997–

IFP V4.0: A Polar-Reformatting Image Formation Processor for Synthetic Aperture Radar

Introduction

IFP V4.0 represents the fourth generation of a very general, multi-purpose image formation software code for spotlight mode synthetic aperture radar (SAR). Using the highly popular and well-known polar reformatting algorithm, the software ingests SAR raw data (phase histories) and auxiliary navigation data and computes phase-coherent complex images. From the beginning, IFP was designed to accommodate data from many different systems and indeed has been successfully employed in data reduction from more than a dozen SAR sources. These include several bistatic SAR systems and a wideband, non-chirped radar in addition to numerous traditional, chirped, monostatic radars.

This document serves several functions. First, a brief history of the gestation and development of IFP over the first three generations is summarized. Second, the enhancements and modifications of this newest version are described. Third, and for the first time, the signal processing equations, with references, are provided in a complete and concise manner, with an aim to documenting the software. And fourth, a user's manual for operating the software is provided.

As will be evident in the following sections, many individuals have contributed along the way to the development of this remarkable and powerful set of software. Phase correction, interferometric pair processing, and terrain mapping software were developed alongside IFP. Together, these four SAR signal processing capabilities represent a generation of SAR R&D accomplishments at Sandia National Laboratories and have had a very significant and permanent national impact.

This document assumes a working knowledge of spotlight-mode SAR image formation and the polar reformatting algorithm. The text by Jakowatz et.al. [1] is a highly recommended primer. Written by the same research team that wrote IFP, many of the concepts and even nomenclature are consistent with this document and the software.

History

Spotlight-mode SAR image formation via the polar reformatting algorithm has its roots in the seminal work of Jack Walker [2].

In the last two years of the 1980's, the polar reformatting algorithm was taken up by a

research group at Sandia (then SNL Org. 0315). This original IFP was conceived with a single purpose in mind: to perform computed image generation from spotlight mode SAR phase histories with two important characteristics. The first defining characteristic was that the software was required to handle extraordinarily large data sets (greater than 50,000 pulses by 10,000 fast time samples) on a typical scientific workstation of the day. The second was that the resultant large complex images had to be formed in one contiguous, phase coherent "patch" so as to support research efforts into the emerging field of SAR interferometry. Neither of these characteristics had been previously demonstrated and both were crucial to the successful prosecution of that department's R&D efforts.

The progenitor of this first IFP, however, was itself a product of two Sandia researchers, Dennis Ghiglia and Gary Mastin, who wrote an especially general polar reformatter code for scientific investigation. Written in FORTRAN77, the code was documented in the technical report SAND90-1793 [3], and later revised in SAND91-0718 [4]. In fact, this progenitor code featured most of the outstanding characteristics to emerge in the IFP code to follow, including:

- out-of-memory processing for large data sets; especially a memory-file-memory block transpose that is particularly efficient.
- ability to produce output imagery in the focus plane, slant plane, or any intermediate plane, efficiently integrated with range interpolation.
- out-of-plane correction to the phase history.
- fast time jitter correction.
- a software architecture designed from the beginning to take advantage of parallel hardware implementations. This feature was also very novel for the time and led to ground-breaking demonstrations on such massively parallel machines as the nCUBE2.
- a highly general model of the auxiliary navigation data and its impact on the signal processing. These fundamental ideas are in large part responsible for the platform generic nature of the code. Parenthetically, a new theory behind this approach was later completely fleshed out in the text ([1]), largely by Jak Jakowatz and Paul Thompson. Not only were the details set to equations, but the entire derivation of spotlight mode image formation was recast in the paradigm of tomographic image reconstruction, forever illuminating the non-platform-specific nature of SAR image formation. The imprint of this approach remains in the code, persisting into the latest generation of IFP4.

Noteworthy in this time sequence, Ghiglia and Mastin also produced a synthetic phase history generator FORTRAN77 code that became, and remains, an invaluable tool in developing, debugging, testing, and certifying all subsequent versions of IFP. For the interested reader, the fundamentals of this idea are developed in the text [1], Appendix D.

Beginning with this fairly complete and functional polar reformatting code, Paul Thompson undertook the task of writing the original version of IFP, which we will heretofore denote IFP1. To the polar reformatting core he added routines to unpack input data, perform phase stabilization, incorporate the new Phase Gradient Autofocus algorithm [5], and detect the output image. He generalized the out-of- memory I/O routines to implement a variable word-length data representation, allowing for the gradual expansion of dynamic range from processing block to processing block without introducing noticable quantization noise at each stage. This allowed for faster intermediate file I/O on the large target data sets.

The all-important step of motion compensation and phase stabilization is fully documented in an internal report [6] and will not be repeated here. However, as the stabilization is actually performed integral to range interpolation in IFP (all versions), those equations will be given in the appropriate section. For information as to the generation of the stabilization coefficients, the reader is referred to [6].

What emerged from this process in March 1991, was a complete end-to-end SAR image formation processing suite. Dubbed IFP and capable of rapid (for the day) processing of large data sets on SUN and Silicon Graphics workstations, it was immediately put to use in first-ever demonstrations of spotlight mode SAR interferometry, massively parallel polar reformatting, high order image autofocus, and other research topics.

Over the next few years, the code was gradually improved in both speed of execution and generality of application. Again largely due to the efforts of Thompson, the interpolation filters were enhanced, mixed radix FFTs were incorporated, and the code was restructured to take advantage of the particularly simple and efficient shared memory parallel processing architectures such as the Silicon Graphics 240-GTX. Gary Mastin ported the code to the Cray XMP architecture, and numerous Sandia researchers got involved in implementation on massively parallel machines: a SIMD CM-2 with 16384 processors, and a MIMD nCUBE 2 with 1024 processors [7]. The workstation implementation and the Cray port were distributed, with support, to numerous national image exploitation organizations such as the Naval Research Laboratory, the Office of Imagery Analysis, and the National Photographic Interpretation Center.

Around 1993, Paul Eichel developed a modification to IFP that allowed its use in the processing of phase histories generated by Sandia's own in-house SAR flying aboard a Twin-Otter aircraft [8]. Unique to this platform, the ability of the radar hardware to effect a pulse-to-pulse modification of its center frequency, chirp starting phase and A/D sample rate resulted in phase histories that were already phase compensated in real-time. However these parameter variations over the aperture required matching modifications to the range interpolator as well as the auxiliary data in order to be properly processed by IFP. The means to unpack the phase history data and generate the proper auxiliary data also had to be developed.

The generalization to bistatic SAR image formation was also being pursued. The fundamental concept of replacing range spheres with ellipsoids and pointing vectors with the

transmitter and receiver pointing vector bisectors was proposed by Eichel, and the necessary frequency scale factor ([1], Eq. 2.76) was worked out by Jakowatz and Thompson. The synthetic phase history generator was modified to generate bistatic data sets, and the generalized IFP was shown to process the data correctly. However, it would be several years before any real bistatic SAR data was so processed.

All of these variations were consolidated into a single, highly versatile code denoted IFP2. The auxiliary data file, orginally denoted by the suffix .aux, was now standardized in a version with suffix .au2 that could accommodate monostatic, bistatic, frequency agile radars, and radars with and without phase stabilized phase histories. IFP2 remained the workhorse for several years and was employed, for example, in processing data from the ERIM P3 L-band SAR, the Global Terrain Mapper (GTM) platform, numerous generations of the Sandia Twin-Otter radars, Joint Stars, and Open Skies, as well as several different bistatic SARs. In many of these applications, although the role of IFP2 was always the same, much effort was required to understand and unpack the (often proprietary) phase histories as well as generate the all-important auxiliary file. Daniel Wahl in particular was instrumental in this role for many of these experiments.

The phase correction routine employed by IFP1 and IFP2 also saw gradual improvement over the years. The original phase correction subroutine was written by Paul Eichel to be replaced by various versions resulting from the work of Daniel Wahl and later Terry Calloway.

In a developmental offshoot, Ireena Erteza completely rewrote IFP2 to implement a parallelized version using the Message Passing Interface (MPI) protocol. This allowed complete portability without rewriting code to any parallel architecture, including heterogeneous and homogeneous systems with distributed processing and memory.

In the late 1990's, Paul Thompson, still the principal scientist charged with maintaining the code, introduced the third generation, IFP3. IFP3 was written to take advantage of the expanded capabilities of FORTRAN90 compilers, particularly allocatable arrays and matrix arithmetic. The multi-processing support was changed to make use of the emerging and standardized OPEN-MP constructs. In the interest of speed, the out-of-memory architecture was changed in favor of in-memory processing, eliminating all intermediate file I/O, as workstation memory sizes had grown to make the former unnecessary. Also, a number of pre-processing tasks were automated in the form of a capable, but rather lengthy and complex shell script called ifp3i. The use of shell scripts to run and control IFP had gradually increased over the years. This trend culminated in ifp3i, which allowed most routine uses of IFP3 to proceed with very little user input.

IFP V4.0: The Latest Version

IFP4 represents a significantly larger break with the past. It was written more with a "clean sheet" mentality than the comparatively evolutionary iterations of the previous versions. In the largest sense, it was meant to address the following issues:

- accommodate in a manner transparent to the user, the numeric endian of the data and the machine on which it is processed.
- generate output files in a standard graphics file format (TIFF 6.0) instead of raw data files.
- provide a command line interface, with no interaction at execution time. This allows for its use in embedded applications.
- written in C and making use of readily available FFTW libraries for portablility. Support for PCs running the Linux OS was a high priority. This is a further reason for the importance of adding numeric endian support.
- eliminate shell script preprocessing. All processing is accomplished in a single executable.
- revise the auxiliary file yet again into a more user-friendly format.
- provide processing equations and a user's manual (this document).

In addition, during the development of the code, it was decided to drop a number of obsolete options with respect to input file types. These file types were no longer in use by their respective platforms, and the need for backward compatibility was judged to be low.

Finally, while multiprocessing support continues with OPEN-MP, the ever increasing size of machine memory meant that the variable word size data representation could be simplified. In IFP4, the data representation is generally single precision floating point complex throughout the processing chain. The only exception is for very large phase histories that originate as one-byte per complex sample. These phase histories are represented internally as two-byte, fixed-point complex values as far as azimuth interpolation, after which the data are converted to single precision floating point complex. All computations regarding geometry were either maintained at, or changed to, double precision floating point.

The employment of TIFF 6.0 output file formats is significant and far reaching. In addition to helping solve the difficulty of endian ambiquity, the extensible tag structure of TIFF file headers allows important auxiliary information to remain attached to the generated imagery. This greatly simplifies the interface between IFP4 and downstream exploitation codes, as the plethora of auxiliary files is no longer needed. A library of user I/O routines has been written to simplify the extraction of image data and auxiliary data from the TIFF

output files. Future revisions or additions to the tag data may be accommodated simply by revising this library: the downstream user codes need only to be relinked. Finally, even without full knowledge of the extended tag structure, the images still conform to the TIFF 6.0 standard and so may be shared with, and used by, third party users for whom no special support is required.

IFP Signal Processing

The IFP executable forms single-patch, phase coherent complex images from (virtually any) spotlight mode SAR phase histories. The following signal processing functions are typically performed, although its behavior can be modified by various command line options:

- The auxiliary file (discussed below) is read in and the image resolution, unaliased image size and dimensions, and required FFT sizes are precomputed. Optionally, the code will compute slow- and fast-time sample trimming to achieve square output image IPR's or a user-specified output resolution. Optionally, the user may specify a particular slow- and fast-time sample set in the interest of coherent pair processing. Optionally, the user can specify output imagery in the focus plane, slant plane or any other plane, and can specify the focus plane. The default focus plane is tangent to the earth ellipsoid at the GRP. Optionally, the user can specify a circumscribed rectangular grid. The user can specify optional output image dimensions. Finally, the user can specify an optional "preview" mode in which the above values are computed, but no phase history data input or processing is initiated.
- The appropriate phase history data is read into memory. During this process, the data is converted from its native numerical representation to complex single precision floating point values (two-byte fixed-point complex for certain phase history types). Also, the endian of the data is converted if it is different from that of the machine on which the processing is being done. On some phase history types, the input data must also be de-multiplexed.
- The phase history undergoes a 1-D interpolation in the range or fast-time direction. The user may optionally specify that this step be skipped for those phase histories already collected by the hardware on a trapezoidal sampling grid (e.g. the SNL Twin-Otter radars). In this document and indeed in the V4.0 code, the term "keystone" is used to refer to such a grid. During this interpolation step the data may also undergo: DC removal, deskewing (for systems wherein the pulse duration is not appreciably larger than the patch size), phase stabilization, and projection into the image plane of choice.
- The data is transposed.
- The data undergoes 1-D interpolation in the cross-range direction.

- The data is transposed.
- Range compression is performed via a mixed-radix FFT. A user-selected aperture weighting function is optionally applied.
- The data is transposed.
- Cross-range compression is performed via a mixed-radix FFT. A user-selected aperture weighting function is optionally applied.
- The image so formed is optionally corrected by means of Phase Gradient Autofocus.
- The complex image is multiplied by the inverse of the antenna beam amplitude pattern.
- The complex image is optionally multi-looked and log detected.
- The complex floating point image and optional detected image are output in TIFF 6.0 format. Auxillary geometric and processing parameters are included in TIFF header tags.

Input and Output Files

The IFP V4.0 executable ingests two files, the raw phase history file and an auxiliary file, and outputs one or two files, the formed complex image and the optional detected image. As mentioned, the output images conform to the TIFF 6.0 specification and so, except for a definition of the parameter tags not normally found in the specification, requires no further documentation. All normal TIFF 6.0 options are supported including, for example, the optional use of compression, tiling, etc.

The input raw phase history file may of course be comprised of various types of data, as different systems do not conform to any particular standard. However, one byte per complex sample, 16-bit integer I/Q, 16-bit magnitude, 16-bit phase, and 32-bit IEEE floating I/Q types are common and supported. Others could be easily accommodated. We have worked with single-rail, real, frequency-offset phase histories, but the conversion is accomplished as a pre-processing step prior to IFP.

The third file type, the input auxiliary file, must conform to a particular standard. In one sense, this file is the glue allowing IFP to adapt to such a wide variety of spotlight mode SARs, both monostatic and bistatic. It captures a canonical set of radar and geometric parameters, all specified in standard engineering units and absolute coordinate systems, required by polar reformatting image formation in its most fundamental and generic sense. The only real "work" involved in adapting a new radar to IFP is the translation of that system's navigation, motion compensation, and waveform particulars into this file type.

An example auxiliary file for IFP V4.0 is depicted in Figure 1. This particular file is from an SNL Twin-Otter type collection. The file is written in ASCII and so has no

```
AUX Version:     V4.0
PH type:         Otter
Date:            May 20, 2003
Geometry:        Monostatic
Sample Grid:     Keystone
Freq Scaling:    Chirp
Bytes/samp:      4
Num of pulses:   3088
Samples/pulse:   2020
Start time:      230076.04000
GRP (ECEF,m):    -1489200.853 -5013844.019 3640799.248
Datum:           WGS-84
Start Freq (Hz): 1.611809263e+10
A/D Freq (Hz):   5.800464000e+07
Gamma (Hz/s^2):  3.341915360e+13
Xmit Pol:        V
Recv Pol:        V
Az BW (rad):     0.042
Elev BW (rad):   0.084
Cal (m/unit):    1.6748e-1

Pulse Data:
#       Tx position (ECEF,m)
        Rx position (ECEF,m)
        time (sec) del_r0 (samples) f0_scale_factor
        C0  C1  C2 (stabilization coefficients)

***********************************************

1       -1494915.494 -5017011.163 3639796.837
        -1494915.494 -5017011.163 3639796.837
        0.000000  0.0  0.996785
        0.0 0.0 0.0

2       -1494915.586 -5017011.074 3639796.926
        -1494915.586 -5017011.074 3639796.926
        0.002000  0.0  0.996787
        0.0 0.0 0.0
        .
        .
        .
```

Figure 1. An example auxiliary file.

inherent numerical format or endian and is instantly readable. At the top are a number of information fields and a few parameters associated with the entire phase history. These are:

AUX Version Must be ≥ 4.0 and < 5.0 to be compatible with IFP V4.0.

PH type A keyword to indicate the platform generating the data and also used to interpret the phase history file.

Date For information purposes.

Geometry Monostatic or Bistatic. This is an informative field only; the code figures everything out from the pulse fields to follow.

Sample Grid Polar or Keystone, to indicate the phase history sampling grid. This is an SNL Twin-Otter collection with keystone sampling.

Freq Scaling None, A/D, or Chirp. Non-frequency agile SARs would be "None". Frequency agile SARs such as the Twin-Otter perform part of the real-time motion compensation by manipulating the fast-time sampling scheme. This can be done via A/D sample rate scaling or chirp rate scaling.

Bytes/samp As the name implies.

Num of pulses The number of slow time pulses in this phase history.

Samples/pulse The number of fast time samples per pulse.

Start time The start time defined in terms of the SAR's time origin. This varies from system to system. For example, in the case of the Twin-Otter, this represents the time in seconds, measured from midnight of the preceeding Sunday, of the first pulse. It is not actually used by IFP for anything.

GRP (ECEF,m) The motion compensation point for this aperture, expressed in Earth-Centered-Earth-Fixed coordinates, meters.

Datum The datum used for all ECEF coordinates. IFP V4.0 currently recognizes only the WGS-84 ellipsoid.

Start Freq (Hz) The frequency represented by the first fast-time sample in Hz. For pulse-to-pulse frequency agile systems, this is defined to be the *desired* or *nominal* start frequency, modified by the f0-scale-factor term in the pulse data. It is also subject to the del-r0 term of the pulse data (see below).

A/D Freq (Hz) The A/D sample rate in Hz.

Gamma (Hz/s^2) The chirp rate in Hz per second squared.

Xmit Pol The transmitter polarization, for information purposes only and optional.

Recv Pol The receiver polarization, for information purposes only and optional.

Az BW (rad) The antenna azimuth beamwidth (3dB, one-way) in radians. Both this entry and the following elevation BW entry are optional. If the .au4 file does not contain these entries, antenna beam compensation will not be performed.

Elev BW (rad) The antenna elevation beamwidth (3dB, one-way) in radians.

Cal (m/unit) Calibration factor in meters per unit. If this optional entry is not present, the calibration computations will not be performed by IFP V4.0.

Other entries of these types may be optionally added (up to a total of 56), but will be ignored by IFP. The fields may actually fall in any order so long as they occur at the beginning of the file and before the "**********" delimiter line.

After the fields of general parameters, the file contains sets of pulse parameters, one set for each slow-time pulse in the phase history. These parameters include:

pulse number Starting from 1.

Tx position The position of the transmitter antenna phase center when the pulse was transmitted. ECEF x,y,z triple, meters, with a precision to one mm.

Rx position The position of the receiver antenna phase center when the pulse was received. Again, ECEF x,y,z triple, meters, with a precision to one mm. Here, the radar is monostatic and slow moving from a relativistic standpoint, so the Tx and Rx positions are identical.

time Time in seconds from pulse 1.

del-r0 Residual time jitter, expressed in units of the A/D sample clock, for motion compensation purposes. Since this Twin-Otter collection was already compensated by the hardware in real-time, the values for this parameter are 0.0 for all pulses. In general, this would not be the case.

f0-scale-factor The ratio of the *nominal* center frequency to the center frequency of *this pulse*. Normally, this value is always 1.0, but varies pulse-to-pulse for frequency agile SARs.

C0,C1,C2 The three phase stabilization coefficients for this pulse, representing the dc, linear, and quadratic fast-time phase terms. These terms should be expressed to 9 significant figures. Again, here they are all 0.0, as the data have been real-time compensated.

This auxiliary file definition is sufficient for most SAR systems. Obviously, it presupposes linear FM chirps are employed for pulse coding. More exotic SARs, such as SNL's non-chirped bistatic SAR using wideband-noise pulses [9], require some pre-processing ahead of IFP.

On the other hand, since position information is provided in absolute coordinates, the images formed by IFP retain the requisite geo-spatial information for subsequent processing software to generate terrain data or orthographic images in true mapping coordinates. That is, the aux file captures sufficient canonical information, and IFP preserves that information, passing it along in the output file tags to support all geo-spatial uses to which that data may be put. No additional support or meta-data files are required.

Initialization

The initialization portion of the IFP V4.0 code provides for command line parsing, array memory allocation, establishment of the focus plane normal, pointing vector computation, and computation of resolutions, image dimensions, fft sizes, and rectangular grid sizes.

Command line parsing

IFP V4.0 is command-line driven as opposed to the interactive nature of IFP3. The command line options (See User's Manual Section) are parsed in a routine called *ifp_clp*. The default processing options are set in this routine, but may be overridden by command line options to use preview mode, use user supplied sets of fast- and slow-time samples, force a user supplied resolution, equalize resolution in range and cross range, choose a focus plane and/or an image plane, choose output image dimensions, enable keystone processing, choose autofocus options, perform range deskew, choose a circumscribed rectangular grid, and choose sidelobe levels and sinc interpolator sizes.

Focus plane normal

The focus plane normal can be specified by the user. If left unspecified, it is determined as the normal to the earth ellipsoid at the GRP given in the auxiliary file. This is computed using the routine *xyz_to_llh* to first convert ECEF x,y,z coordinates to geodetic latitude, longitude, and height. This computation follows the iterative approach given in [10], Eq. 6.28 through Eq. 6.36. Throughout IFP4, the datum used is the WGS-84 ellipsoid with semi-major axis, a=6378137.0 m, and flattening, f=1.0/290.257223563. If the GRP is also left unspecified (as for example with a synthetic phase history), the focus plane normal is assigned to be (0,0,1).

Pointing vectors

Unlike previous versions, IFP4 treats every phase history as if it is bistatic. The bistatic equations reduce to the monostatic in the case of co-located transmitter and receiver posi-

tions. The pointing vectors and the "beta" array are computed in the routine *aux_point*. For each pulse, we have:

$$\begin{aligned}
\vec{p}_t &= \vec{r}_{tx} - \vec{r}_{grp} \\
\vec{p}_r &= \vec{r}_{rx} - \vec{r}_{grp} \\
\vec{p} &= \left\{ \frac{|\vec{p}_t| + |\vec{p}_r|}{2} \right\} \left(\frac{\vec{p}_t}{|\vec{p}_t|} + \frac{\vec{p}_r}{|\vec{p}_r|} \right) / \left| \frac{\vec{p}_t}{|\vec{p}_t|} + \frac{\vec{p}_r}{|\vec{p}_r|} \right| \qquad (1)
\end{aligned}$$

In these equations, $\vec{r}_{tx}$ is the transmit position (ECEF, meters), $\vec{r}_{rx}$ is the receive position, $\vec{r}_{grp}$ is the ground reference (motion compensation) point, and $\vec{p}$ is the computed pointing vector.

Coordinate axes

The determination of slant plane, focus plane, and image plane coordinate axes generally follows the procedure given in [1], Appendix C. The precise procedure is given below. Note that all vectors are assumed to be expressed in ECEF coordinates.

Given the focus plane normal, $\bar{z}$, we approximate the focus plane x- and y- axes by setting y to be the bisector of the aperture angle in the focus plane:

$$\begin{aligned}
\hat{v}_{f1} &= \frac{\vec{p}_1 - (\vec{p}_1 \cdot \bar{z})\bar{z}}{|\vec{p}_1 - (\vec{p}_1 \cdot \bar{z})\bar{z}|} \\
\hat{v}_{fN} &= \frac{\vec{p}_N - (\vec{p}_N \cdot \bar{z})\bar{z}}{|\vec{p}_N - (\vec{p}_N \cdot \bar{z})\bar{z}|} \\
\bar{y} &= \frac{\hat{v}_{f1} + \hat{v}_{fN}}{\left|\hat{v}_{f1} + \hat{v}_{fN}\right|} \qquad (2) \\
\bar{x} &= \bar{y} \times \bar{z} \qquad (3)
\end{aligned}$$

Within IFP4, the full phase history contains *ph_npulse* pulses in the range [0,*ph_npulse*-1]. An image can be formed from any contiguous subset of these pulses. We label *n_pulse_offset* the number of pulses at the beginning of the aperture to be skipped, and *n_pulse* the number of pulses to process, i.e. the set [*n_pulse_offset*,*n_pulse_offset*+*n_pulse*-1] is actually processed into an image. The subscripts 1 and N in these equations correspond to the first and last pulses of this set.

Next, we establish a "slant plane". Of course, we do not assume the phase history was actually collected in a plane. However, we fit a plane to the physical trajectory of the platform by means of the so-called 20% - 80% approximation. That is, we find the pointing

vectors that correspond to 20% and 80% of the aperture angle subtended in the focus plane. These pointing vectors then define the slant plane normal. Let:

$$\theta_1 = cos^{-1}(\hat{v}_{f1} \cdot \bar{x}) \quad (4)$$

$$\theta_N = cos^{-1}(\hat{v}_{fN} \cdot \bar{x}) \quad (5)$$

And:

$$\theta_{20} = 0.8\theta_1 + 0.2\theta_N \quad (6)$$

$$\theta_{80} = 0.2\theta_1 + 0.8\theta_N \quad (7)$$

Then, for every pulse, we find:

$$\hat{v}_{fi} = \frac{\vec{p}_i - (\vec{p}_i \cdot \bar{z})\bar{z}}{|\vec{p}_i - (\vec{p}_i \cdot \bar{z})\bar{z}|}$$

$$\theta_i = cos^{-1}(\hat{v}_{fi} \cdot \bar{x}) \quad (8)$$

and determine the pulse indices for which θ_i is closest to θ_{20} and θ_{80}. Let's label these indices i_{20} and i_{80}, respectively. We then define the slant plane normal as:

$$\hat{z} = \frac{\vec{p}_{i_{20}} \times \vec{p}_{i_{80}}}{|\vec{p}_{i_{20}} \times \vec{p}_{i_{80}}|} \quad (9)$$

If the inner product of $\hat{z}$ so calculated with $\bar{z}$ is negative, $\hat{z}$ is multiplied by -1 so that it points "up". Finally, the *line-of-sight* vector, $\vec{r}_{los}$ (which will determine the final image orientation), is defined as:

$$\vec{r}_{los} = 0.5\,(\vec{p}_{i_{20}} + \vec{p}_{i_{80}}) \quad (10)$$

Having determined a slant plane normal and an LOS vector, we can complete the slant plane coordinate system as:

$$\hat{y} = \frac{\vec{r}_{los}}{|\vec{r}_{los}|}$$

$$\hat{x} = \hat{y} \times \hat{z} \quad (11)$$

Finally, we determine the image plane coordinate axes. First the image plane normal, $\tilde{z}$, is set to the focus plane normal, $\bar{z}$ (default), or the slant plane normal, $\hat{z}$ (user option), or a user specified vector. Then:

$$\tilde{y} = \frac{\vec{r}_{los} - \frac{\vec{r}_{los} \cdot \tilde{z}}{\tilde{z} \cdot \tilde{z}}\tilde{z}}{\left|\vec{r}_{los} - \frac{\vec{r}_{los} \cdot \tilde{z}}{\tilde{z} \cdot \tilde{z}}\tilde{z}\right|}$$

$$\tilde{x} = \tilde{y} \times \tilde{z} \quad (12)$$

The resultant image will therefore be in the plane defined by $\tilde{z}$ with a range axis $\tilde{y}$ and cross-range axis $\tilde{x}$. The azimuth orientation of the image with respect to true north is:

$$azimuth = tan^{-1}\left(\frac{\tilde{x}_z}{\tilde{y}_z}\right) \tag{13}$$

The resolution loop

The routine that computes the coordinate axes given in the preceeding section (*get_axes*) is called in the resolution loop. This loop adjusts the slow-time pulses and fast-time samples in an iterative fashion to achieve some desired image resolution. During each pass through the loop, the coordinate axes are recomputed (since they depend on the set of pulses chosen), the region of support of the chosen phase history in the image plane is determined, and the resultant range and cross-range resolutions computed. The loop exits when the desired outcome is achieved.

After the axes are computed, the phase history region of support in the image plane must be found. We describe the procedure for inscribed rectangular grids. First, the *proscal* values for every pulse in the chosen set are computed. While this computation is similar to that described in [4], it has been generalized to accommodate bistatic geometries and pulse-to-pulse frequency agile radars. For every pulse, we find:

$$\begin{aligned} \vec{p'}_i &= \vec{p}_i - \frac{\vec{p}_i \cdot \tilde{z}}{\tilde{z} \cdot \bar{z}}\bar{z} \\ proscal_i &= \frac{|\vec{p}_i|}{\left|\vec{p'}_i\right|} * \frac{fscale_i}{cos(\beta_i/2)} \end{aligned} \tag{14}$$

Here, $\vec{p}_i$ is the pointing vector for the i^{th} pulse, and $\vec{p'}_i$ is a vector in the image plane that results from projecting $\vec{p}_i$ to the focus plane and then finding the intersection of the projection line with the image plane. See [4], pp. 7-10 or [1], pp. 187-189 for details. The value $proscal_i$ is the ratio of the length of $\vec{p}_i$ to the length of $\vec{p'}_i$, adjusted by $fscale_i$ and β_i. The first term accomplishes both out-of-plane correction and projection to the image plane ([1], p. 188). The factor $fscale_i$ accommodates the instantaeous center frequency of this pulse for frequency agile radars, and $cos(\beta_i/2)$ is the frequency scaling term for nonzero bistatic angles ([1], Eq. 2.76).

In like fashion, the scale factor *vpmag* is computed. This quantity is the inverse of *proscal*, computed using the LOS vector:

$$\begin{aligned} \vec{r'}_{los} &= \vec{r}_{los} - \frac{\vec{r}_{los} \cdot \tilde{z}}{\tilde{z} \cdot \bar{z}}\bar{z} \\ vpmag &= \frac{\left|\vec{r'}_{los}\right|}{|\vec{r}_{los}|} * \frac{cos(\beta_{N/2}/2)}{fscale_{N/2}} \end{aligned} \tag{15}$$

where the frequency scale factor and the bistatic angle of the middle pulse is used to approximate those values for the LOS vector. This is necessary since the LOS vector does not actually correspond to any of the physical pointing vectors. The value *vpmag* represents the overall range scale factor from the nominal slant plane to the image plane.

The angle a given pointing vector makes with the image plane y-axis, measured in the image plane, is called ϕ_i:

$$\phi_i = -\tan^{-1}\left(\frac{\vec{p}_i \cdot \tilde{x}}{\vec{p}_i \cdot \tilde{y}}\right) \tag{16}$$

The *aperture angle* subtended by the processed phase history, as measured in the image plane is therefore:

$$\theta = |\phi_N - \phi_1| \tag{17}$$

And the peak *out-of-plane angle* (from which the depth of focus is calculated) is given by:

$$\Delta\psi = max\left[sin^{-1}\left(\frac{\vec{p}_i \cdot \hat{z}}{|\vec{p}_i|}\right)\right] - min\left[sin^{-1}\left(\frac{\vec{p}_i \cdot \hat{z}}{|\vec{p}_i|}\right)\right] \tag{18}$$

With these preliminaries accomplished, we are now in a position to compute the resulting image domain range and azimuth resolution. If we process *n_samp* fast time samples (offset *n_samp_offset* from the first sample of each pulse), then the "range" size of the aperture, projected to the image plane, measured in inverse meters is:

$$\Delta Y = (n\_samp - 1) * \frac{2\gamma}{cf_{ad}} * vpmag \tag{19}$$

where γ is the radar chirp rate and f_{ad} is the A/D sample rate. The term $\frac{2\gamma}{cf_{ad}}$ is the spatial frequency (m^{-1}) per fast-time sample of the input data and *vpmag* scales it to the image plane. The "cross-range" aperture size is:

$$\Delta X = \left(\frac{f_{st} f_{ad}}{\gamma} + n\_samp\_offset\right) * \frac{2\gamma}{cf_{ad}} * vpmag * (2tan(\theta/2)) \tag{20}$$

Here, the term $\frac{f_{st} f_{ad}}{\gamma}$, where f_{st} is the chirp start frequency, is the inner polar radius (in fast-time samples) of the phase history.

Assuming a weighting function is applied for sidelobe control with an IPR broadening factor of B_F, the resulting image domain range and cross-range resolutions are therefore:

$$\begin{aligned} \rho_y &= \frac{B\_F}{\Delta Y} \\ \rho_x &= \frac{B\_F}{\Delta X} \end{aligned} \tag{21}$$

The remaining logic in the *get_resolution* routine iteratively adjusts the slow-time pulses and fast-time samples used in the above equations to obtain a desired image resolution. The loop is always executed at least once; it will be executed several times if the user specifies equalized range and azimuth resolution or specifies a specific resolution is to be obtained.

Rectangular Grid dimensions, FFT sizes, and Image dimensions

The final parameters that are computed in the initialization section of the code are the rectagular grid dimensions, the FFT lengths to be used for range and azimuth compression, and the output image dimensions. These parameters are actually computed in the reverse of this order, since the user may optionally specify a (smaller) output image size.

We first find the maximum unaliased image size that can be generated from this phase history. IFP4 defaults to an zero-pad value of 0.3 (set in the ifp.h include file). The range and azimuth interpolator impulse response functions are of length *2*nzero + 1*, where the default *nzero* = 8 (user option). Therefore the maximum, unaliased image dimensions are:

$$\begin{aligned} n\_img\_rg\_max &= 1.3 * \frac{nzero}{nzero+1} * n\_samp \\ n\_img\_az\_max &= 1.3 * \frac{nzero}{nzero+1} * n\_pulse \end{aligned} \tag{22}$$

The user may specify output dimensions smaller than these. Larger values will be upper-bounded by a limit 1.5 times these values (to allow for some degree of phase history up-sampling).

The FFT lengths are chosen in a routine that chooses values greater than or equal to the corresponding image dimensions times the factor $\frac{nzero+1}{nzero}$. The FFT lengths chosen may actually be larger than this, because the lengths are required to be products of terms, each of which are 2 raised to a small prime. This is to allow for efficient mixed radix FFT's. The rectangular grid dimensions are then simply the corresponding FFT sizes divided by the one plus the zero-pad value:

$$\begin{aligned} n\_rect\_rg &= n\_fft\_rg/1.3 \\ n\_rect\_az &= n\_fft\_az/1.3 \end{aligned} \tag{23}$$

In the case keystone processing is specified by the user, assuming the phase history was so generated, the FFT sizes are retained but the zero-pad value is adjusted so that the range rectangular grid dimension equals the number of fast-time samples, *n_samp*.

A few other quantities may now be calculated. The spacing of samples of the rectangular grid in the image plane (the output of the range and azimuth interpolation process) is given by:

$$\begin{aligned} rect\_ss\_rg &= \Delta Y/(n\_rect\_rg-1) \\ rect\_ss\_az &= \Delta X/(n\_rect\_az-1) \end{aligned} \tag{24}$$

where the units are inverse meters per sample. The image scale factors are therefore:

$$\begin{aligned} ss\_rg &= 1.0/\left[rect\_ss\_rg * (n\_fft\_rg - 1)\right] \\ ss\_az &= 1.0/\left[rect\_ss\_az * (n\_fft\_az - 1)\right] \end{aligned} \tag{25}$$

in meters per pixel. Finally, the interpolator resample ratios are:

$$\begin{aligned} resamp\_rg &= \frac{n\_rect\_rg - 1}{\Delta Y / \left(\frac{2\gamma}{cf_{ad}} * vpmag\right)} \\ &= \frac{\frac{2\gamma}{cf_{ad}} * vpmag}{rect\_ss\_rg} \\ resamp\_az &= \frac{n\_rect\_az - 1}{npulse - 1} \end{aligned} \tag{26}$$

If the user has specified the preview option, the code prints out a summary of the parameters calculated to this point and exits. Otherwise, the phase history is read in and processing proceeds to the range interpolator. As the phase history is read into memory, a data endian correction is performed if necessary, and each complex sample is converted from its native representation into four-byte floating point I,Q complex. The buffer into which it is read is sized to accommodate the input data or the output of the range interpolator, whichever is larger. This allows the range interpolator to function *in-place*. Due to the needs of *range bandwidth extension* and *range deskew* (discussed in the next section), all fast-time samples of a given pulse are read into the buffer, regardless of the size *n_samp*. However, only the set of pulses chosen by the resolution loop (*n_pulse* begining with *n_pulse_offset*) are read in.

Range Interpolation

In IFP, quite a few operations are performed in the range interpolation routine. These include dc bias removal, phase stabilization, range deskew, aperture weighting for circumscribed grids, and finally the range interpolation itself. These processing steps are performed independently on each radar pulse, thereby allowing a natural parallelization over multiple threads. Like IFP3, IFP4 uses OPEN-MP parallelization constructs.

The first task in the routine is to expand the domain of fast-time samples on which these processing steps are applied to accommodate highly squinted geometries and range deskew. If, as a result of the resolution loop computations, all of the available fast-time samples are to be processed anyway, this expansion procedure will, of course, have no effect. However, if a reduced range resolution is desired (and therefore $n\_samp < ph\_nsamp$), these two expansions are appropriate.

Consider a highly squinted imaging geometry with the image to be formed in the focus plane. Unless the radar is frequency agile pulse-to-pulse, the region of support of the

aperture function, projected to the image (focus) plane has a pronouced "droop" (see [1], Fig. 3.56). Because of the way in which the rectangular grid is constructed, a subset of the available fast-time samples does not completely "fill up" the rectangular grid. It is therefore prudent to interpolate more fast-time samples to the rectangular grid, if they are available. Consequently, the range interpolator code extends the fast-time sample set selected in the resolution loop by as much as 25% on each end. We call this range bandwidth extension:

$$\begin{aligned} n_{ext} &= 0.25 * n\_samp \\ n\_interp\_offset &= max[0, n\_samp\_offset - n_{ext}] \\ n\_interp &= min[ph\_nsamp, n\_samp\_offset + n\_samp + n_{ext}] \\ &\quad - n\_interp\_offset \end{aligned} \tag{27}$$

Fast-time sample extension for deskew is similar. Because the deskew operation effectively *translates* the data in fast-time, we should perform the deskewing operation on a larger set of fast-time samples than what is required for a given resolution. We find:

$$\begin{aligned} max\_deskew\_shift &= \frac{f_{ad}^2}{2\gamma} \\ n\_deskew\_offset &= max[0, n\_interp\_offset - max\_deskew\_shift] \\ n\_deskew &= min[ph\_nsamp, n\_interp\_offset + n\_interp + max\_deskew\_shift] \\ &\quad - n\_deskew\_offset \end{aligned} \tag{28}$$

Expressed in terms of the sample index, the deskew operation has a domain of $[n\_deskew\_offset, n\_deskew\_offset + n\_deskew]$ and a range of $[n\_interp\_offset, n\_interp\_offset + n\_interp]$. This is followed by the range interpolator, which has a domain of $[n\_interp\_offset, n\_interp\_offset + n\_interp]$ and a range of $[0, n\_rect\_rg]$.

DC Bias Removal

The first processing step is to remove any dc bias in the phase history data. For each pulse, $i \in [n\_pulse\_offset, n\_pulse\_offset + n\_pulse]$, we compute:

$$\bar{x}_i = \frac{1}{n\_deskew} \sum_n x_i(n) \tag{29}$$

where $x_i(n)$ represents the complex fast-time sample at index n. These per-pulse bias values are filtered as:

$$\bar{x} = 0.99\bar{x} + 0.01\bar{x}_i \tag{30}$$

And the resultant filtered bias is removed:

$$\tilde{x}_i(n) = x_i(n) - \bar{x} \tag{31}$$

Phase Stablization

IFP V4.0 continues with the motion compensation philosophy of earlier versions of IFP. The notion is that the phase stabilization that must be applied to any spotlight mode phase history for the purpose of motion compensation may be distilled into the following equation:

$$\vartheta(i,n) = C_0(i) + C_1(i)n + C_2(i)n^2 \tag{32}$$

This equation describes the phase correction, $e^{j\vartheta(i,n)}$, that multiplies the n^{th} fast-time sample of the i^{th} pulse. The three coefficients, $C_0(i)$, $C_1(i)$, and $C_2(i)$, are a function of the pulse index i and represent the phase (in radians), the frequency (in radians per sample), and the chirp (in radians per sample squared), respectively, that must be applied to the pulse.

IFP assumes the three per-pulse coefficients are computed externally and stored in the auxiliary AU4 file. This helps insulate IFP from the hardware and system details of particular SAR platforms. We note that various systems may also require a per-pulse time-jitter correction to the fast-time samples as well. This correction is stored in the AU4 file as del_r0 values and is applied at the range interpolation step.

In the code, Equation 32 is implemented with a recursion. We rewrite the equation as:

$$\begin{aligned} \vartheta(i,n) &= \vartheta(i,n-1) + d(n-1) \\ d(n) &= d(n-1) + 2C_2(i) \end{aligned} \tag{33}$$

with initial values:

$$\begin{aligned} \vartheta(i,0) &= C_0(i) \\ d(0) &= C_1(i) + C_2(i) \end{aligned} \tag{34}$$

The principal values of phase $\vartheta(i,n)$ computed in this way are quantized over the interval $[0, 2\pi]$ and used to index a lookup table of correction terms $e^{j\vartheta(i,n)}$.

Range Deskew

Range deskew, if selected, is also accomplished in the range interpolation routine. Although easy to describe, this step is computationally expensive. Each pulse of the phase history must be forward Fourier transformed, multiplied by a correction vector and inverse transformed. If $N_{fft} \geq n\_deskew$ represents the size of the FFTs used, then the correction vector is:

$$DSK(n) = \frac{1}{N_{fft}} e^{-j\frac{\pi f_{ad}^2 n^2}{\gamma N_{fft}^2}} \tag{35}$$

Range Interpolation

The range interpolator resamples the polar sampled data onto a trapezoidal grid ([1], pp. 133-135). In IFP, however, this step also accomplishes *out of plane correction* and *projection into the image plane*. These latter two functions are attained at essentially no additional computational cost by means of a very clever interpolator design. Also, the interpolator filter coefficients are precomputed and stored in a lookup table, resulting in an absolute minimum of computations.

To see how this is accomplished, we start with the interpolator equation (see [1], Eq. 3.47 and Figure 3.28):

$$x'_m = \frac{T}{T'} \sum_n x_n \frac{sin\{\pi(t_m - t_n)/(\beta T')\}}{\pi(t_m - t_n)/(\beta T')} \left[0.5 + 0.5cos\{\pi(t_m - t_n)/(\beta T' N_z)\}\right] \tag{36}$$

Here, x'_m is the output sample at time t_m, the x_n are the input samples at times t_n, T and T' are the input and output sample intervals, respectively, N_z is the interpolator filter length in number of zero crossings, and β is the *bandwidth reduction factor*. N_z is a user selectable option, but defaults to 8. We use the following empirical formula for β:

$$\beta = 1.005 + 0.49/N_z + 0.544/N_z^2 \tag{37}$$

In Equation 36, we see each output sample is constructed as a weighted sum of input samples. The interpolator filter is a Hanning weighted, truncated sinc function. Rather than computing the filter coefficients anew for each input and output sample pair, a coefficient table is constructed by quantizing the time difference $t_m - t_n$. In general, the projected input sample x_n falls somewhere between an adjacent pair of output sample points. Let us label as t_q the output sample point *before* time t_n, so: $0 \leq (t_n - t_q) < T'$. We decompose the time difference $t_m - t_n$ into two components:

$$\begin{aligned} t_m - t_n &= (t_m - t_q) - (t_n - t_q) \\ &= (j_{n,m} T') - (t_n - t_q) \end{aligned} \tag{38}$$

where $j_{n,m} = (t_m - t_q)/T'$ is the number of output sample intervals from t_q to t_m, an integer. We then quantize the time $t_n - t_q$ as:

$$t_n - t_q \approx i_n \frac{T'}{N_f} \tag{39}$$

i.e. an output sample interval is quantized into N_f fractions. Note that both $j_{n,m}$ and i_n are integers. The former depends on both the input sample time t_n and the output sample time t_m but the latter depends only on t_n, hence the subscripts.

So we have:

$$t_m - t_n = j_{n,m} T' - i_n \frac{T'}{N_f} \tag{40}$$

We may now build a table of coefficients as:

$$TABL(i,j) = \frac{sin\{\pi(j-i/N_f)/\beta\}}{\pi(j-i/N_f)/\beta}\left[0.5+0.5cos\{\pi(j-i/N_f)/(\beta N_z)\}\right] \tag{41}$$

where $j \in [-(N_z-1), N_z]$ and $i \in [0, N_f]$. Returning to the interpolation Eq. 36, we have:

$$x'_m = \frac{T}{T'}\sum_n x_n * TABL(i_n, j_{n,m}) \tag{42}$$

where the summation is restricted to those values of n for which $j_{n,m}$ fall in the range $[-(N_z-1), N_z]$.

Having built the table, all that remains is to be able to determine the indices $j_{n,m}$ and i_n at execution time. More precisely, for every input sample x_n, we must find the corresponding time t_n in terms of the rectangular grid (output) sample locations, t_m. The easiest way to find a general expression for this unknown is through the tomographic formulation of spotlight mode SAR. In the following, the reader should refer to [1], Figures 2.19 and 3.49.

Consider the n^{th} fast-time input sample, x_n. In the spatial frequency domain, this sample lies a distance of:

$$r_n = r0 + n \tag{43}$$

polar samples from the origin, where:

$$r0 = \frac{f_{st} f_{ad}}{\gamma} - del_{r0}(i) + n\_interp\_offset \tag{44}$$

is the polar radius of the first fast-time sample in samples. This equation takes into account the fast-time jitter term $del\_r0(i)$ for the current pulse as well as n_interp_offset. Each fast-time sample represents $\frac{2\gamma}{cf_{ad}}$ inverse meters, so r_n expressed in inverse meters is:

$$R_n = r_n \frac{2\gamma}{cf_{ad}} \tag{45}$$

When projected into the image plane (in a direction normal to the focus plane), the *Y component* of this distance becomes:

$$(R'_n)_Y = \frac{R_n}{proscal_i} cos(\phi_i) \tag{46}$$

where the subscript i refers to the pulse number to which this sample belongs. The scale factor $proscal_i$ achieves the projection of this pulse into the image plane and the term $cos(\phi_i)$ yields the Y component. Rearranging Equation 26 to get the output sample spacing:

$$rect\_ss\_rg = \frac{\frac{2\gamma}{cf_{ad}} * vpmag}{resamp\_rg} \tag{47}$$

in inverse meters per sample. Combining the above four equations, we have the (Y) distance of the projected input sample, expressed in terms of output sample intervals, as:

$$\begin{aligned}(r'_n)_Y &= (R'_n)_Y / rect\_ss\_rg \\ &= \left\{ \frac{r_n \frac{2\gamma}{cf_{ad}}}{proscal_i} cos(\phi_i) \right\} / \left\{ \frac{\frac{2\gamma}{cf_{ad}} * vpmag}{resamp\_rg} \right\} \\ &= (r0+n) \frac{cos(\phi_i) * resamp\_rg}{proscal_i * vpmag} \end{aligned} \tag{48}$$

We pause to comment on Equation 48. This equation gives us the position, *in output sample intervals*, of the input sample projected into the output image plane. This remarkable equation involves only basic radar parameters (γ, f_{st}, f_{ad}), imaging geometry ($proscal_i$, $vpmag$, ϕ_i), and user selected parameters (n_interp_offset, $resamp\_rg$. In other words it works with essentially any spotlight mode SAR, whether monostatic, bistatic, frequency-agile, any frequency, chirp rate, etc. In one step, it accomplishes projection of the phase history into the desired image plane, accommodates squint, and corrects for out of plane motion. It (and a similar location equation to be developed for the cross-range interpolator) is where the power and generality of IFP can most clearly be seen.

We are now in a position to find $j_{n,m}$ and i_n. Let us pick the origin of the rectangular array in range (the first sample of the output rectangular array) to be where a hypothetical polar sample at radius $r0$ fast-time samples from the origin and along a hypothetical pulse aligned with the slant plane y-axis, $\hat{y}$, is projected into the image plane:

$$(r'_0)_Y = r0 * resamp\_rg \tag{49}$$

This can be seen easily from Equation 48 by noting that $n = 0$, $\phi_i = 0$, $proscal_i * vpmag = 1$ for such a point. Therefore, we have:

$$\begin{aligned} t_n &= T' * \left[(r'_n)_Y - (r'_0)_Y \right] \\ &= T' * \left[(r0+n) \frac{cos(\phi_i) * resamp\_rg}{proscal_i * vpmag} \right. \\ &\quad \left. - r0 * resamp\_rg \right] \end{aligned} \tag{50}$$

We simplify this equation as

$$t_n = T' * y_n \tag{51}$$

where:

$$\begin{aligned} y_n &= (r0+n) * a - d_0 \\ a &= \frac{cos(\phi_i) * resamp\_rg}{proscal_i * vpmag} \\ d_0 &= r0 * resamp\_rg \end{aligned} \tag{52}$$

Equation 51 yields t_n in inverse meters from the origin of the rectangular array, and Equation 52 gives that distance in output samples. For the historical record, we note that Equation 52 is the essence of the routine *LOCATE_R()* from the original implementation of [3]. From Equation 39, we have:

$$\frac{t_n}{T'} - \frac{t_q}{T'} \approx \frac{i_n}{N_f} \tag{53}$$

But, by definition, t_q is a sample point in the output array. Hence, $\frac{t_q}{T'}$ is an *integer*. In fact, it is the largest integer not greater than $\frac{t_n}{T'}$. Therefore:

$$\begin{aligned} i_n &= N_f * \left[\frac{t_n}{T'} - \frac{t_q}{T'}\right] \\ &= N_f * \left[\frac{t_n}{T'} - floor\{\frac{t_n}{T'}\}\right] \\ &= N_f * [y_n - floor\{y_n\}] \end{aligned} \tag{54}$$

Finally, $j_{n,m}$ is the number of output sample intervals from t_q to t_m, so we also have:

$$j_{n,m} = m - floor\{y_n\} \tag{55}$$

The range interpolator may be efficiently implemented as follows. Given an input radar pulse i, we compute the constants $r0$, a, and $d0$ for Equation 52. Then for every input sample x_n of that pulse, we find y_n from Equation 52, i_n from Equation 54 and $j_{n,m}$ from Equation 55. Finally, we perform the sum, Equation 42. We note that the sample ratio T/T' in Equation 42 is simply equal to the coefficient a from Equation 52. This ratio is a function of pulse number, but is constant for a given pulse.

Cross-Range Interpolation

Cross-range interpolation completes the polar reformatting of the phase history onto a rectangular sampling grid. The "extra" processing steps of the range interpolation routine, namely dc bias removal, phase stabilization, and deskew, have no analog in the cross-range interpolator routine.

The interpolator filter table is built exactly as before (Equation 41), but of course we need to derive new expressions for $j_{n,m}$ and i_n. As before, the subscript n refers to the input sample index, and the subscript m to the output sample index. In this context, the n^{th} input sample comes from the n^{th} processed *pulse*, and the output samples are from the rectangular grid in the X direction.

After range interpolation, the data is resampled onto a keystone grid. The Y coordinate of the p^{th} row of this grid is simply:

$$(R'_p)_Y = T'(r'_0)_Y + (p * rect\_ss\_rg)$$

$$
\begin{aligned}
&= r1 * \frac{2\gamma}{cf_{st}} * vpmag + (p * rect\_ss\_rg) \\
&= grid1 + (p * rect\_ss\_rg) \qquad (56)
\end{aligned}
$$

Remember, $(r'_0)_Y$ from Equation 49 is expressed in output samples, so we multiply by $T' = \frac{2\gamma}{cf_{st}} * vpmag/resamp\_rg$ to find the zero grid line in inverse meters. This value is called *grid1* in the code.

Since the angle ϕ_n is defined to be the angle the n^{th} pulse makes with the image plane y-axis, the n^{th} cross-range grid sample of the p^{th} keystone row has an X coordinate of:

$$
\begin{aligned}
(R'_{p,n})_X &= (R'_p)_Y * tan\{\phi_n\} \\
&= [grid1 + (p * rect\_ss\_rg)] * tan\{\phi_n\} \qquad (57)
\end{aligned}
$$

in inverse meters. If we take the origin of the rectangular grid in X to be $-n\_rect_{az}/2$, then the above input sample is at a location:

$$
\begin{aligned}
x_n &= \frac{(R'_{p,n})_X}{rect\_ss\_az} + n\_rect_az/2 \\
&= \frac{[grid1 + (p * rect\_ss\_rg)] * tan\{\phi_n\}}{rect\_ss\_az} + n\_rect_{az}/2 \qquad (58)
\end{aligned}
$$

expressed in *samples*. In like fashion to Equation 52, this expression for x_n gives us the location of the n^{th} input sample to the interpolator in terms of the output samples. And just like Equations 54 and 55, we have:

$$i_n = N_f * [x_n - floor\{x_n\}] \qquad (59)$$

$$j_{n,m} = m - floor\{x_n\} \qquad (60)$$

These last three equations, together with the table lookup filter coefficients are used to implement the cross-range interpolator. In cross-range interpolation, the input sample spacing is not necessarily constant, even along a given keystone row, so a sample ratio cannot be used for normalization. Instead, we normalize by:

$$x'_m = \frac{\sum_n x_n * TABL(i_n, j_{n,m})}{\sum_n TABL(i_n, j_{n,m})} \qquad (61)$$

Transpose

In polar reformatting image formation, the two-dimensional data array must be transposed three times: between range and azimuth interpolation, between azimuth interpolation and range compression, and between range and azimuth compression. IFP V4.0 performs this

step using an out-of-place block transpose between two in-memory arrays. Because this is not an in-place operation, one of these transposes always represents the point at which the program has allocated the most memory.

The program implements a fairly conventional block transpose. That is, two-dimensional blocks of data are extricated from the larger input data array, transposed, and then inserted into the proper place in the output data array. Perhaps the only unusual aspect of this implementation is the fact that the dimensions of the block used are parameters set at compilation time. Since the efficiency of a block transpose is a function of the cache size and implementation on a given machine, the block dimensions can be fine-tuned to maximize overall efficiency of the transpose. We have a diagnostic code, cornerTurn4, that empirically determines the optimum block dimensions for a given machine or architecture. The block parameters are then set to these values in the ifp.h include file before compilation.

Range and Azimuth Compression

The compression routine, called for both range and azimuth compression, performs one-dimensional Fast Fourier Transforms on rows of the input data array. The routine is written as an out-of-place operation with arguments pointing to the input and output data arrays. For both range and final azimuth compression, the routine is called with both pointers set to a single array. It is thus used as an in-place operation. However, a preliminary azimuth compression is used to generate a temporary image that functions as input to the phase error correction routine. Since the range-compressed data must be saved for the final azimuth compression step, this preliminary compression is performed out-of-place. Depending on the image size, the temporary image may be either full-size or reduced in size. If reduced, the reduction factor in both the range and cross-range dimensions is determined by a parameter, PGA_REDUCTION_FACTOR, in ifp.h.

In addition to performing the FFT-based compression, this routine also applies an aperture weighting function for sidelobe suppression, and optionally multiplies each row of the input data by a phase error vector for phase correction. This error vector is obtained from the phase error estimation routine. The FFT routines used by IFP V4.0 are called from the widely available third-party FFTW library, compiled as single precision floating point.

Phase Error Estimation

The phase error vector applied during final azimuth compression is estimated in a separate routine implementing the Phase Gradient Autofocus (PGA) algorithm. This algorithm has been extensively documented in the literature, and will not be elaborated upon here [11].

IFP4 is executed from the command line. A built-in help menu may be accessed by typing:

```
% ifp4
```

or:

```
% ifp4 -h
```

In order to process a dataset, IFP4 requires one argument and accepts numerous options. Generally, the argument corresponds to the *rootname* of the input files. That is, given two input files with names, *rootname*.au4 and *rootname*.phs, the command:

```
% ifp4 rootname
```

will cause the dataset to be formed into an image with the default parameters (discussed below). The source code provides a provision for incorporating a pre-processor. For example, if the argument is the name of an *etpm* file (with extension), the etpm pre-processor will be executed:

```
% ifp4 file_name.etpm
```

This will result in the formation of a .au4 and .phs file from the .etpm file, followed by the execution of IFP4. The choice of pre-processors is determined by the extension of the argument file.

The operation of IFP4 may be altered through the use of command-line options. Options are invoked by a - followed by an option keyword and possible arguments. Only the first letter of any keyword need be typed. Options may be specified in any order, but any arguments must immediately follow their corresponding keyword. The options supported are:

-preview Instructs IFP4 to compute image sizes, parameters, geospatial parameters, etc. without actually going to the trouble (and time) of forming an image.

-use_samples #pulses #offset #samples #offset All four arguments are required. This option instructs IFP4 to employ only **#pulses** number of pulses from the phase history, **#offset** from the beginning, in its processing. Likewise, **#samples** fast-time samples, **#offset** from the first, are employed. This allows *aperture trimming* in fast and slow time. The default is to use all the samples.

-resolution #res_in_meters Form image to **#res_in_meters** meters IPR in both range and cross-range, if the phase history supports it. The default is whatever resolution results from the chosen fast- and slow-time input sample set. This option implies the -equalize_resolution option.

-equalize_resolution IFP4 will reduce either the fast- or slow-time sample set to achieve the best possible, but equal, IPR resolutions in range and cross-range, within the constraints imposed by the -use_samples option.

-focus_plane x y z Normally, IPF4 uses as a focus plane the plane that is tangent to the Earth (WGS-84) ellipsoid at the GRP. The user may override this default by specifying a focus plane normal in ECEF coordinates.

-image_plane slant, x y z The default choice of image plane normal is the focus plane. If the user specifies **slant**, the image will be formed in the slant plane. Or the user may specify an arbitrary image plane normal, **x y z**, in ECEF coordinates.

-output_image_dimensions #n_az #n_rg The output image will be formed with **#n_az** cross-range samples and **#n_rg** range samples. These can be larger or smaller than the default values. The default values are computed from input parameters and results in an image just slightly smaller than nyquist in both directions.

-keystone_enabled Keystone processing will be used if the input phase history supports it, bypassing range interpolation. The default is to have keystone processing disabled. Setting this argument disables deskew.

-autofocus_disabled Autofocus is disabled with this argument. The default is enabled.

-write_phase_error_vector If specified and if autofocus is enabled, the estimated phase error vector is written to a file with extension .phe.

-make_detected_image #n_ds #db_range #db_rms This option causes a multi-looked, log-detected image to be computed from the single-look complex image and written to an output file. The **#n_ds** parameter calls for a (ds X ds) multi-look and downsample. The log-detected image will have a total dynamic range of **#db_range** dB, and a minimum value **#db_rms** dB below the image rms value. Thus, the **#db_range** parameter functions as the contrast control, and the **#db_rms** as the brightness.

-deskew If specified, deskew will be applied in range processing. Deskew is disabled by default.

-xscribe The rectangular array will be adjusted to circumscribe the phase history in spatial frequencies. This is not recommended except for very unusual circumstances. The default is to inscribe the rectangular array.

-sidelobe_ratio #ratio_in_dB Usually, aperture weighting ensuring -40dB peak sidelobes is employed. The user may override this by specifying a different peak sidelobe ratio.

-zeros #zeros_interpolator The interpolators have weighted sinc function impulse responses of length 8 zero-crossings on each side of the main lobe. The user may override the default length with this option.

As an example, the command:

```
% ifp4 file_name.etpm -r 0.4 -o 12000 12000 -a -d -m 5 50 20
```

will process the specified etpm file to a resolution of 0.4 meters in range and cross-range and output an image of size 12000 X 12000 samples. No autofocus will be applied, and range deskew will be applied. A log-detected image will be produced with a 5x5 multi-look downsample, a dynamic range of 50dB, and minimum value 20dB below the image rms. The command:

```
% ifp4 file_name.etpm -u 20000 120 9800 300 -s slant -p
```

will employ 20000 slow-time pulses, offset 120 from the beginning, and 9800 fast-time samples, offset by 300; image in the slant plane; with preview only (no actual image formation).

Conclusion

It is hoped that this document fills an important niche, linking the theoretical development of spotlight mode SAR using the tomographic paradigm developed in the text by Jakowatz, et. al. [1] with the software implementation of IFP V4.0. Many important details are not readily accessible from either the text or the code. By setting out all of the relevent signal processing equations and cross referencing the corresponding development in the text, the transition from theory to practice should be much more clear to the interested researcher or radar engineer. The latest software version, V4.0, implements these equations exactly and in a most transparent fashion.

References

[1] Jakowatz, Wahl, Eichel, Thompson, and Ghiglia, *Spotlight-mode Synthetic Aperture Radar: A Signal Processing Approach*, Klewer Academic Publishers, 1995.

[2] Jack L. Walker, "Range-doppler imaging of rotating objects," *IEEE Transactions AES*, vol. AES-16, 1980.

[3] Gary Mastin and Dennis Ghiglia, "A research-oriented spotlight synthetic aperture radar polar reformatter," Technical report SAND90-1793, Sandia National Laboratories, Albuquerque, New Mexico 87185 and Livermore, California 94550, October 1990.

[4] Gary Mastin and Dennis Ghiglia, "An enhanced spotlight synthetic aperture radar polar reformatter," Technical report SAND91-0718, Sandia National Laboratories, Albuquerque, New Mexico 87185 and Livermore, California 94550, March 1992.

[5] Eichel, Jakowatz, and Ghiglia, "Speckle processing method for synthetic aperture radar phase correction," *Optics Letters*, vol. 14, no. 1, 1989.

[6] Paul A. Thompson, "Phase history stabilization," Internal technical report, Sandia National Laboratories.

[7] Plimpton, Mastin, and Ghiglia, "Synthetic aperture radar image processing on parallel supercomputers," *SuperComputing 91*, 1991.

[8] Wells, Sorensen, Doerry, and Remund, "Developments in sar and ifsar systems and technologies at sandia national laboratories," *IEEE Aerospace Conference, Big Sky, MT*, March 2003.

[9] Cordaro, Eichel, Bickel, and Burns, "Bistatic interferometric sar and gmti results," *50th Annual Tri-Service Radar Symposium*, June 2004.

[10] Alfred Leick, *GPS Satellite Surveying*, John Wiley & Sons, 1995.

[11] Wahl, Eichel, Ghiglia, and Jakowatz, "Phase gradient autofocus - a robust tool for high resolution sar phase correction," *IEEE Transactions AES*, vol. AES-30, no. 3, 1994.

DISTRIBUTION:

1 D. C. Ghiglia
Chief Scientist, Vexcel Corp.
1690 38th Street
Boulder, CO 80301

1 G. A. Mastin
Lockheed Martin IS&S
P.O. Box 85, MS 5011
Litchfield Park, AZ 85340-0085

1 MS 1207
C. V. Jakowatz, 5937

1 MS 1207
T. M. Calloway, 5937

1 MS 1207
N. E. Doren, 5937

10 MS 1207
P. H. Eichel, 5937

1 MS 1207
I. A. Erteza, 5937

1 MS 1207
D. E. Wahl, 5937

1 MS 1207
D. A. Yocky, 5937

1 MS 0529
B. L. Remund, 5340

1 MS 0519
B. L. Burns, 5340

1 MS 0519
W. H. Hensley, 5342

1 MS 0519
T. P. Bielek, 5342

1 MS 1330
A. W. Doerry, 5342

1 MS 0519
D. W. Harmony, 5342

1 MS 0519
D. G. Thompson, 5342

1 MS 1330
K. W. Sorensen, 5345

1 MS 1330
D. F. Dubbert, 5345

1 MS 1330
S. M. Becker, 5348

1 MS 1330
S. M. Devonshire, 5348

1 MS 0519
L. M. Wells, 5354

1 MS 0519
D. L. Bickel, 5354

1 MS 0519
J. T. Cordaro, 5354

1 MS 0519
J. M. Delaurentis, 5354

3 MS 9018
Central Technical Files, 8945-1

2 MS 0899
Technical Library, 9616