March 2017

Time Domain SAR Processing with GPUs for Airborne Platforms

Dustin Lagoy
University of Massachusetts Amherst

TIME DOMAIN SAR PROCESSING WITH GPUS FOR AIRBORNE PLATFORMS

A Thesis Presented

by

DUSTIN LAGOY

Submitted to the Graduate School of the
University of Massachusetts Amherst in partial fulfillment
of the requirements for the degree of

MASTER OF SCIENCE

February, 2017

Electrical and Computer Engineering

TIME DOMAIN SAR PROCESSING WITH GPUS FOR AIRBORNE PLATFORMS

A Thesis Presented

by

DUSTIN LAGOY

Approved as to style and content by:

Paul R. Siqueira, Chair

Patrick A. Kelly, Member

Ramakrishna Janaswamy, Member

C.V. Hollot, Department Chair
Electrical and Computer Engineering

Now, here*, you see, it takes all the running* you *can do, to keep in the same place. If you want to get somewhere else, you must run at least twice as fast as that!*

— Lewis Carrol
Through the Looking-Glass

ACKNOWLEDGEMENTS

I am especially thankful for the endless support and guidance provided by my advisor, Paul Siqueira. Professors Patrick Kelly and Ramakrishna Janaswamy deserve thanks for being part of my thesis committee and providing useful feedback on my work.

All of the past and current students and staff of MIRSL who have worked on the S- and Ka-band radar systems have my gratitude. They have undoubtedly put in more hours than I making these systems what they are. This work, and my time at MIRSL in general, would have been a lot less interesting without them.

Thanks go out to everyone on the UAVSAR team at JPL for their work and guidance during my time there. Brian Hawkins deserves special thanks for his mentoring, support, for answering all my questions despite his busy schedule and for his work on the Angler processor.

This section would not be complete without thanking professor Thomas Millette of Mount Holyoke College. Without his abundant helpfullness and flexibility, none of our radar systems would ever have made it off the ground.

Thank you to everyone else in my life for putting up with me and keeping me sane.

Finally, thank you, the reader. This thesis is written for you as much as for me, and it is my sincerest hope that you will be able to extract something useful from it.

ABSTRACT

TIME DOMAIN SAR PROCESSING WITH GPUS FOR AIRBORNE PLATFORMS

FEBRUARY, 2017

DUSTIN LAGOY, B.S., UNIVERSITY OF MASSACHUSETTS AMHERST

M.S., UNIVERSITY OF MASSACHUSETTS AMHERST

Directed by: Professor Paul R. Siqueira

A time-domain backprojection processor for airborne synthetic aperture radar (SAR) has been developed at the University of Massachusetts' Microwave Remote Sensing Lab (MIRSL). The aim of this work is to produce a SAR processor capable of addressing the motion compensation issues faced by frequency-domain processing algorithms, in order to create well focused SAR imagery suitable for interferometry. The time-domain backprojection algorithm inherently compensates for non-linear platform motion, dependent on the availability of accurate measurements of the motion. The implementation must manage the relatively high computational burden of the backprojection algorithm, which is done using modern graphics processing units (GPUs), programmed with NVIDIA's CUDA language. An implementation of the Non-Equispaced Fast Fourier Transform (NERFFT) is used to enable efficient and accurate range interpolation as a critical step of the processing. The phase of time-domain processed imagery is different than that of frequency-domain imagery, leading to a potentially different approach to interferometry. This general purpose SAR processor is designed to work with a novel, dual-frequency S- and Ka-band radar system

developed at MIRSL as well as the UAVSAR instrument developed by NASA's Jet Propulsion Laboratory. These instruments represent a wide range of SAR system parameters, ensuring the ability of the processor to work with most any airborne SAR. Results are presented from these two systems, showing good performance of the processor itself.

TABLE OF CONTENTS

LIST OF TABLES

Table **Page**

LIST OF FIGURES

Figure **Page**

CHAPTER 1

INTRODUCTION

Synthetic aperture radar (SAR) interferometers have long been used to measure various spatially distributed phenomena on the Earth's surface and in space. Such phenomena include large and small scale topography, properties of natural land cover of many types and snow and ice, among others. These radar systems are capable of making widespread measurements which can be repeated with relatively high frequency, enabling scientific studies that rely on such a measurement scheme. Generally, SARs will either be operated from an airborne or spaceborne platform, depending on a campaign's budget and science requirements. Besides the great difference in altitude, airborne systems must also contend with the much less regular motion of the platform due to the atmosphere.

To generate useful imagery of the phenomenon under investigation, a SAR system must compensate for the finite size of its antenna beam, which is typically on the order of degrees. Without such a compensation, the resolution of a SAR system would be limited in the azimuth direction, to something on the order of kilometers. To improve this resolution a SAR system will "synthesize" a large antenna array by coherently adding the recorded returns of individual pulses measured during the motion of the platform. Typically, this coherent addition is achieved through one of several SAR processing algorithms which operate in the frequency-domain. While such algorithms enable efficient generation of high resolution imagery, they are limited by the difficulty of compensating for non-ideal motion of the platform. For airborne platforms, where non-ideal motion can be significant, other options may be preferred.

There exist SAR processing algorithms that instead operate in the time-domain which are generally simpler and which will inherently compensate for any known, non-ideal motion. The main reason why these have seen little use historically is due to their high computational complexity when compared to frequency-domain algorithms.

Due to the desire to achieve more accurate imagery with airborne platforms, and the increase in general and parallel computing resources, time-domain algorithms are starting to be used with modern airborne SAR systems. There are two SAR interferometers under development at the University of Massachusetts' Microwave Remote Sensing Lab; one at S-band (3.2GHz) and one at Ka-band (35GHz) [5, 10]. Both of these systems operate on a small, low-altitude aircraft. NASA's Jet Propulsion Laboratory (JPL) has developed, and operates, an L-band SAR system, which currently uses motion-compensated frequency-domain algorithms to perform its SAR processing [18]. Both systems can be seen in Figure 1.1.

The S-band and Ka-band radar systems are designed to operate simultaneously on the same platform, allowing for novel, dual-frequency, interferometric data acquisition. Due to the order of magnitude difference in wavelength, the scattering and penetration depth of the two radars should differ significantly for many terrain types. Since the radars operate simultaneously, the dual-frequency system is able to show these different characteristics without temporal or spatial decorrelation.

This thesis describes the development and implementation of a time-domain processor, designed to generate high resolution imagery with both UMass radars as well as with NASA's UAVSAR instrument. In addition, results are presented with each of the mentioned radar systems and using simulated data, including interferometric results with the dual-frequency system. Chapter 2 lays out the signal model for a generic airborne synthetic aperture radar. In Chapter 3 a comparison of several SAR processing algorithms is made, with an emphasis on backprojection; the algorithm used in this implementation. Chapter 4 describes fundamental differences between

(a) KaSI.

(b) UAVSAR.

Figure 1.1: The two relevant radar systems. KaSI, the UMass system, is a combination of the S- and Ka-band radars.

time and frequency-domain algorithms with respect to interferometry. Details regarding the implementation of the processor are found in Chapter 5, including a discussion of general purpose computing with graphics processing units (GPUs), and why it is applicable to time-domain backprojection. Relevant coordinate systems are presented in Chapter 6, including mathematical descriptions of conversions between them. A step-by-step description of the written processor is given in Chapter 7. Finally, Chapter 8 presents results obtained using this processor.

1.1 Review of past work

Synthetic aperture radar imaging is a mature technology, dating back to the optical SAR processing of the 1960s [7]. Not long after, SAR processing began to be processed digitally, as it is today, where modern processors for airborne and spaceborne systems typically use frequency-domain algorithms like the Range-Doppler, Chirp-Scaling or ωK algorithms [7, 19, 4]. A detailed review of frequency-domain SAR processing is beyond the scope of this document. In order to perform SAR processing in the frequency-domain, it is required to assume that the radar platform flies linearly[1] during the entire data acquisition. This assumption does not often well reflect the true flight path, especially for turbulent airborne systems. It is possible to extend these processing algorithms to correct errors due to non-linear motion, by adding additional motion compensation steps to the processor [19, 15]. This adds additional computational burden and complexity to the processor.

As described by both Cumming *et al.* and Soumekh [7, 19], an alternative is to use time-domain based algorithms, which do not inherently assume a linear flight path. Soumekh [19] gives a brief discussion of the time-domain correlation and backprojection algorithms though the focus of the book is on frequency-domain processing.

[1]Here linear indicates a flight track which can be represented by a straight line in some Cartesian coordinate system.

The simplicity of the algorithms is clear in the two page discussion, where the main drawbacks of the algorithms, namely the high computational burden, is presented.

Ribalta [16] derives in detail the time-domain correlation and backprojection algorithms for FMCW radar systems (which often operate on small, low-altitude aircraft and are especially prone to non-linear motion). The derivation includes detail regarding the start-stop approximation and how it simplifies the correlation algorithm to the backprojection algorithm. In addition, Ribalta describes a corrected backprojection algorithm, which models motion of the platform during chirp transmission as linear, to reduce the impact of the start-stop approximation for FMCW systems (which generally have longer pulse lengths). The treatment is mostly theoretical, but does include simulation results comparing the three described algorithms. Ribalta's work is an important reference as a starting point for a basic backprojection processor, but does not cover pulsed based systems.

Stringham *et al.* [20] have developed a time-domain backprojection processor, also for FMCW systems. An additional term is added to Ribalta's corrected backprojection algorithm, using a least-squares approximation of second order terms, to improve the correction for systems with very wide bandwidths and long chirp lengths. Additionally, the processor is written to run on graphics processing units (GPUs) to reduce computation time. An overview of the implementation is presented for stripmap SAR systems, including a discussion of compensation for variable synthetic aperture size due to platform motion. Additionally, Fourmont's [9] NERFFT interpolation method is used for the critical interpolation step of the backprojection algorithm. Processed results are presented from a single FMCW system. Stringham describes some important implementation details relevant to SAR backprojection processing and is, in fact, very similar to the work presented here. In some places of Stringham's work, the presented work emphasizes computational speed over accuracy, most notably in the interpolation step, though this is minor. Stringham's processor is not designed

to work with other types of radar systems and the description does not include the additional output products presented here.

Capozzoli [3] gives a good description of many interpolation methods relevant to backprojection processing, from nearest neighbor, to cubic, to Knab as well as the NERFFT. Implementation details are discussed, using GPU hardware. A detailed comparison of accuracy and computational time is presented in Capozzoli's work, using both numerical and experimental data. Capozzoli puts forth a good case for the NERFFT interpolation method, originally presented by Fourmont [9] for tomography. The NERFFT is shown to give by far the most accurate results with only minor differences in processing time.

As far as this author is aware, Duersch *et al.* [8] presents the only discussion regarding SAR interferometry using backprojected data. In comparison, there is a wealth of information about SAR interferometry for more traditional, frequency-based, processing, for example as presented by Rosen *et al.* [17]. In the work by Duersch *et al.* a theoretical derivation of height estimates for scatterers for cross track interferometers processed with the backprojection algorithm is presented. The result, including some simplifying assumptions of the system geometry, is a relation between the interferometric phase and scatterer height. What is most interesting is the comparison with traditional interferometry which shows not only a different relation between the phase and height but also different sensitivities to system parameters (geometric or otherwise). It is shown that, in general, interferometry performed with backprojected imagery is less sensitive to geometric errors, but more sensitive to phase errors. This article is of particular interest as the radar systems in use in this document are designed for interferometry.

There have been recent developments in backprojection SAR processing using GPU hardware. The work presented here continues this development by implementing a GPU-based SAR backprojection processor designed to work with a wide range

of radar systems, including both pulsed and FMCW systems. Additionally interferometric results are presented using the backprojection processor and imagery collected from the dual-frequency system.

CHAPTER 2

SAR SIGNAL MODEL

Linear-frequency modulated radar systems are commonly used on airborne and spaceborne platforms. These systems transmit a pulse whose frequency changes linearly with time. In such a configuration, there are two main dimensions of importance. There is the azimuth dimension corresponding to the flight track of the platform, and the range dimension which is measured orthogonal to the azimuth dimension. In Figure 2.1 these are represented by the y-axis, and either r_{slant} or r_{ground} respectively. Since the imaged region exists on, or at least near, the surface, analyzing the system can generally be reduced to these two dimensions. When operating over a flat surface, range and azimuth can be directly mapped to the surface assuming the flight track is at an altitude, h. In this case, the azimuth dimension is directly translated to the surface, and the range is mapped to the ground using

$$r_{\text{ground}} = \sqrt{r_{\text{slant}}^2 - h^2} = r_{\text{slant}} \sin(\theta_i) \tag{2.1}$$

where the slant range is the orthogonal range from the platform. A similar mapping can be done for a more generic curved surface.

Due to the ability to pulse compress a linear FM signal, the inherent resolution in the range direction will be $\Delta\rho = \frac{c}{2BW}$, where c is the speed of light and BW is the bandwidth of the FM chirp [7]. Commonly, this resolution is on the order of meters. In the azimuth direction, the resolution is limited by the size of the antenna beam which can be anywhere from tens of meters to kilometers depending on the

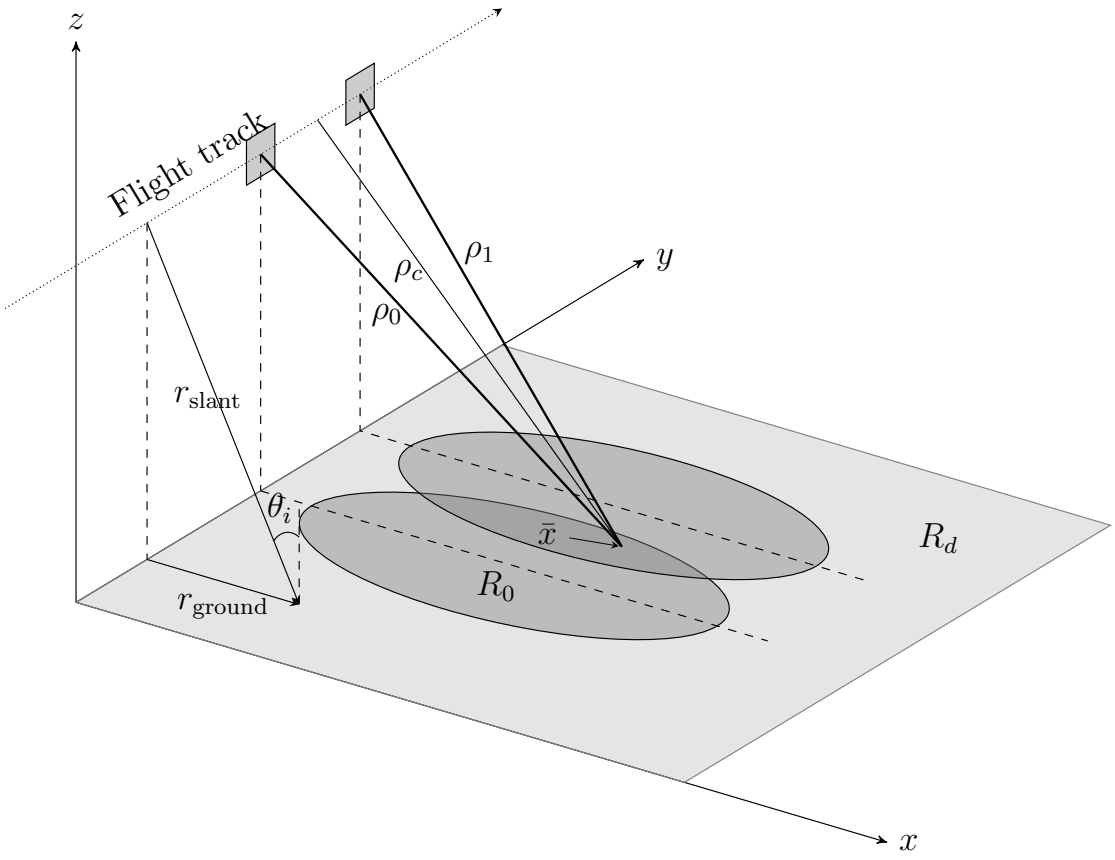


Figure 2.1: Generic SAR geometry.

antenna and platform characteristics. While it is relatively easy to increase the range resolution by increasing the system bandwidth, it is difficult to make an antenna beam fine enough to achieve a similar resolution.

Synthetic aperture radar is a technique which overcomes this limiting azimuth resolution by taking advantage of the motion of the platform and the continuous data acquisition of the radar. Given a fast enough pulse repetition rate, there will be many pulses that illuminate a given target as the platform flies past. From the antenna's point of view, the target will be traversing the width of the beam at a range determined by the hyperbolic equation

$$\rho = \sqrt{\rho_c^2 + v_0^2 t_{az}^2} \tag{2.2}$$

where ρ_c is the range at the closest approach, v_0 is the nominal velocity of the platform and t_{az} is the "slow" time used to represent the distance traveled in the azimuth direction relative to the point of closest approach [7]. By exploiting this known geometry, and the properties of the signal, it is possible to add the contributions of each pulse together in such a way as to synthesize a much larger antenna. Each pulse which illuminates the target can be seen as a single sample of a large antenna array whose size is determined by the width of the beam (meaning it is possible to synthesize a larger array with a wider antenna beam). It can be shown that with the phase of these separate pulses properly accounted for, such a system is able to produce a theoretical azimuth resolution of approximately half the azimuth dimension of the antenna [7]. Hence, a one meter long antenna can achieve a theoretical resolution of fifty centimeters.

2.1 Platform Attitude

Relevant to the discussion of most synthetic aperture radar systems is a note about the attitude, or spatial orientation, of the radar platform. Especially for airborne

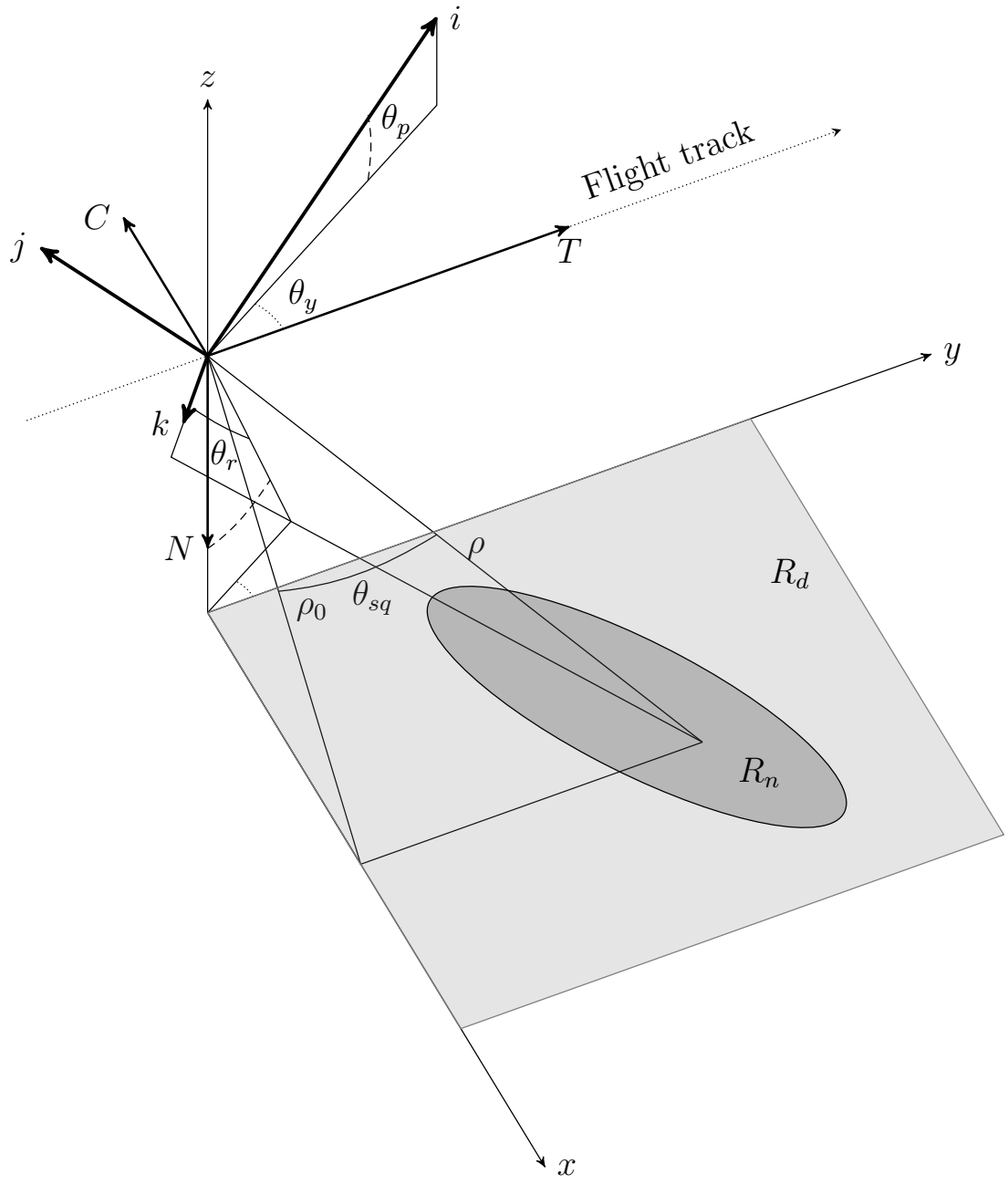


Figure 2.2: SAR attitude.

systems, the orientation of the platform will not always be aligned with its direction of travel. Figure 2.2 shows a depiction of this. The x-, y- and z-axes and flight track are defined the same as in Figure 2.1 but here the current platform location is on the z-axis. The TCN, or Track, Cross-Track, Nadir, coordinate system is defined relative to the platform motion such that T is aligned with the platform velocity vector. The ijk coordinate system represents how the platform is oriented relative to its motion. For an aircraft i, would be aligned with the craft's nose and j with its left wing. The attitude angles θ_y, θ_p and θ_r, or yaw, pitch and roll, are Euler angles used to represent the platform's orientation (i.e. ijk relative to TCN). For SAR systems, a value of interest is the platform's squint angle, θ_{sq}, representing the degree to which the antenna beam is offset on the ground. Note that the squint angle will depend not only on the yaw and pitch attitude angles, but also the platform height and the range at which the squint is measured.

2.2 Pulsed Radar Signal Model

The following signal-model derivation is based on the work of Carrara and Ribalta [4, 16]. A generic linear-frequency modulated radar chirp transmitted by either an FMCW or pulsed radar system will have the form

$$s_{tx}(n,t) = w_r(t)\cos\left(2\pi f_0 t + \pi K_r t^2\right) \tag{2.3}$$

where f_0 is the radar center frequency and K_r is the linear chirp rate. The chirp rate, $K_r = \frac{BW}{T}$, corresponds to a linear chirp of duration T, and bandwidth BW. The window function

$$w_r(t) = \text{rect}\left(\frac{t}{T}\right) = \begin{cases} 1 & |t| < \frac{T}{2} \\ 0 & \text{else} \end{cases} \tag{2.4}$$

defines the duration of the chirp. The index, n, is used to distinguish individual pulses, each of which uses its own local reference time, where $t = 0$ is the center of the chirp.

The received waveform, for either case, will be an amplitude-modulated delayed replica of the transmitted waveform

$$\begin{aligned} s_{rx}(n,t) &= \int_{R_n} \sigma(\bar{x}) A_{rx}(\bar{x},n,t) s_{tx}\left(t - t_0(\bar{x},n,t)\right) dV \\ &= \int_{R_n} \sigma(\bar{x}) A_{rx}(\bar{x},n,t) w_r\left(t - t_0(\bar{x},n,t)\right) \\ &\qquad \times \cos\left(2\pi f_0\left(t - t_0(\bar{x},n,t)\right) + \pi K_r\left(t - t_0(\bar{x},n,t)\right)^2\right) dV. \end{aligned} \tag{2.5}$$

Here, R_n is the illuminated region, where $\bar{x}$ is the three-dimensional coordinates of a point within the region, σ is the reflectivity of the target at $\bar{x}$ and $A_{rx}(\bar{x},n,t)$ is the system amplitude factor due to the antenna pattern and system gain. The volume integral, $\int_{R_n} dV$, represents the returns from all scatterers illuminated by the pulse n. This would generally include individual point and volume scatterers. The integral could be replaced by a surface integral in simplified cases. The delay is defined to be

$$t_0(\bar{x},n,t) = \frac{2\rho(\bar{x},\bar{p}_{n,t})}{c} \tag{2.6}$$

where $\rho(\bar{x},\bar{p}_{n,t})$ is the distance between the target at $\bar{x}$ and the platform at $\bar{p}_{n,t}$. This assumes that the transmit and receive antennas are co-located. Note that the antenna pattern for a given point can be assumed to be constant during the duration of the pulse such that

$$A_{rx}(\bar{x},n,t) = A_{rx}(\bar{x},n,0) = A_{rx}(\bar{x},n). \tag{2.7}$$

For pulsed radars, the received waveform can be downconverted to a baseband signal mathematically, as

$$s_{if}(n,t) = s_I(n,t) + j s_Q(n,t) \tag{2.8}$$

where

$$s_I(n,t) = \text{LPF}\left\{s_{rx}(n,t)\cos(2\pi f_0 t)\right\} \tag{2.9}$$

and

$$s_Q(n,t) = \text{LPF}\left\{s_{rx}(n,t)\cos\left(2\pi f_0 t + \frac{\pi}{2}\right)\right\}. \tag{2.10}$$

This results in (via Euler's formula)

$$\begin{aligned} s_{if}(n,t) = \int_{R_n} & \sigma(\bar{x})A_{rx}(\bar{x},n,t)w_r\left(t - t_0(\bar{x},n,t)\right) \\ & e^{-j2\pi f_0 t_0(\bar{x},n,t)} e^{j\pi K_r\left(t - t_0(\bar{x},n,t)\right)^2} dV \end{aligned} \tag{2.11}$$

where we have ignored a factor of $\frac{1}{2}$ which is instead included in the system gain.

2.3 FMCW Radar Signal Model

Some radar systems (generally these are FMCW systems) compress the signal in range by mixing the received signal with a copy of the transmitted signal, as in

$$s_{if}(n,t) = \text{LPF}\left\{s_{rx}(n,t)\, s_{tx}(t - t_d)\right\} \tag{2.12}$$

where we define $s_{tx}(t) \triangleq s_{tx}(0,t) = s_{tx}(n,t)$ which is true of most radar systems. The delay, t_d, will generally be zero for FMCW systems, but can be non-zero if this method is used for pulsed systems. This will have the form

$$\begin{aligned} s_{if}(n,t) = \text{LPF}\Bigg\{ \int_{R_n} & \sigma(\bar{x})A_{rx}(\bar{x},n)w_r\left(t - t_0(\bar{x},n,t)\right) \\ & \times \cos\left(2\pi f_0\left(t - t_0(\bar{x},n,t)\right) + \pi K_r\left(t - t_0(\bar{x},n,t)\right)^2\right) \\ & \times w_r(t - t_d)\cos\left(2\pi f_0(t - t_d) + \pi K_r(t - t_d)^2\right) dV \Bigg\} \end{aligned} \tag{2.13}$$

which can be simplified to

$$s_{if}(n,t) = \frac{1}{2}\int_{R_n} \sigma(\bar{x})A_{rx}(\bar{x},n)w_r\left(t - t_0(\bar{x},n,t)\right)\cos\left(\phi(\bar{x},n,t)\right) dV. \tag{2.14}$$

In (2.14) the phase term, $\phi(\bar{x}, n, t)$ can be written as

$$\begin{aligned}\phi\left(\bar{x}, n, t\right) = &- 2\pi\left(t_0\left(\bar{x}, n, t\right) - t_d\right)\left(f_0 + K_r t\right) \\ &+ \pi K_r\left(\left(t_0\left(\bar{x}, n, t\right)\right)^2 - t_d^2\right).\end{aligned} \tag{2.15}$$

The second term in the phase of s_{if} is the quadratic phase error.

2.4 Pulse Compression

Pulse compression of the signal in range will either start from (2.11) for most pulsed systems, or from (2.14) for FMCW systems. In both cases, the platform is assumed to be stationary during the transmission of the pulse (known as the *start-stop* or *stop-and-go* approximation) so that

$$\rho(\bar{x}, \bar{p}_{n,t}) = \rho(\bar{x}, \bar{p}_{n,0}) = \rho(\bar{x}, \bar{p}_n). \tag{2.16}$$

or

$$t_0\left(\bar{x}, n, t\right) = t_0\left(\bar{x}, n\right). \tag{2.17}$$

This approximation can be significant, especially for FMCW systems with longer pulse durations.

2.4.1 Pulsed Radar

For pulsed radars, the pulse compression can be implemented as a convolution with a reference function applied in time- or frequency-domains. Starting from (2.11), and using the time-reversed, complex conjugate of the baseband transmit signal as the reference function

$$s_{\text{ref}}\left(t\right) = w_r\left(t\right) e^{-j\pi K_r t^2} \tag{2.18}$$

the pulse compression is described as

$$s_{pc}(n,t) = \int_{-\infty}^{\infty} s_{if}(n,u)\, s_{\text{ref}}(t-u)\, du. \tag{2.19}$$

This has the form

$$\begin{aligned} s_{pc}(n,t) = \int_{-\infty}^{\infty} \int_{R_n} & \sigma(\bar{x}) A_{rx}(\bar{x},n,t) w_r\left(u - t_0(\bar{x},n)\right) \\ & \times e^{-j2\pi f_0 t_0(\bar{x},n)} e^{j\pi K_r (u - t_0(\bar{x},n))^2} dV \\ & \times w_r(t-u)\, e^{-j\pi K_r (t-u)^2} du. \end{aligned} \tag{2.20}$$

or

$$\begin{aligned} s_{pc}(n,t) = \int_{R_n} & \sigma(\bar{x}) A_{rx}(\bar{x},n,t) e^{-j2\pi f_0 t_0(\bar{x},n)} e^{j\pi K_r \left(t_0(\bar{x},n)^2 - t^2\right)} \\ & \times \int_{-\infty}^{\infty} w_r\left(u - t_0(\bar{x},n)\right) w_r(t-u) \\ & \times e^{j2\pi K_r u (t - t_0(\bar{x},n))} du dV. \end{aligned} \tag{2.21}$$

In this case, the size of the reference function and the signal are the same (as determined by the window functions w_r). [1] Here the inner integral can be split into two parts, one where the reference function is to the left of the signal ($t > t_0$) and one where it is to the right ($t < t_0$), giving

[1] In [7] it is shown that the case where the sizes of the reference function and the signal are different leads to the same result.

$$
\begin{aligned}
w_r\left(t-t_0\left(\bar{x},n\right)+\frac{T}{2}\right)\int_{-\infty}^{\infty}\operatorname{rect}\left(\frac{u-\frac{t+t_0(\bar{x},n)}{2}}{t-t_0\left(\bar{x},n\right)+T}\right)\\
\times e^{j2\pi uK_r(t-t_0(\bar{x},n))}du\\
+w_r\left(t-t_0\left(\bar{x},n\right)-\frac{T}{2}\right)\int_{-\infty}^{\infty}\operatorname{rect}\left(\frac{u-\frac{t+t_0(\bar{x},n)}{2}}{t_0\left(\bar{x},n\right)-t+T}\right)\\
\times e^{j2\pi uK_r(t-t_0(\bar{x},n))}du. \qquad (2.22)
\end{aligned}
$$

These integrals can be solved as Fourier transforms using $f=-K_rt$ as the frequency variable. This results in

$$
\begin{aligned}
&w_r\left(t-t_0\left(\bar{x},n\right)+\frac{T}{2}\right)\left(t-t_0\left(\bar{x},n\right)+T\right)e^{-j2\pi(-K_rt+K_rt_0(\bar{x},n))\frac{t+t_0(\bar{x},n)}{2}}\\
&\times\operatorname{sinc}\left(\left(t-t_0\left(\bar{x},n\right)+T\right)\left(-K_rt+K_rt_0\left(\bar{x},n\right)\right)\right)\\
&+w_r\left(t-t_0\left(\bar{x},n\right)-\frac{T}{2}\right)\left(t_0\left(\bar{x},n\right)-t+T\right)e^{-j2\pi(-K_rt+K_rt_0(\bar{x},n))\frac{t+t_0(\bar{x},n)}{2}}\\
&\times\operatorname{sinc}\left(\left(t_0\left(\bar{x},n\right)-t+T\right)\left(-K_rt+K_rt_0\left(\bar{x},n\right)\right)\right). \qquad (2.23)
\end{aligned}
$$

The exponential term in both sums are identical, and cancel with the term in the outer integral of (2.21). For radars with a large time-bandwidth product (TBP = T^2K_r), this can be well approximated as [7]

$$
\begin{aligned}
s_{pc}\left(n,t\right)\approx\int_{R_n}\sigma(\bar{x})A_{rx}(\bar{x},n)e^{-j2\pi f_0t_0(\bar{x},n)}\\
\times T\operatorname{sinc}\left(TK_r\left(t-t_0\left(\bar{x},n\right)\right)\right)dV \qquad (2.24)
\end{aligned}
$$

which leads to a time-range mapping of $t=\frac{2\rho}{c}$.

In practice, the received signal is sampled to give

$$
s_{if}[n,t_m]=s_{if}(n,mT_s) \qquad (2.25)
$$

where T_s is the sampling period and t_m represents the discrete time index. In solving using the Fourier transform, the frequency variable $f_k = -K_r t_m$ would be used in the discrete case. The discrete Fourier transform is defined to give the relation $f_k = \frac{k}{T}$ or $f_k = \frac{k-M}{T}$ depending on how the discrete system is set up. If set up to give a positive range relation the sinc term transforms from $T\,\mathrm{sinc}(T(-K_r t + K_r t_0\,(\bar{x}, n)))$ to $T\,\mathrm{sinc}\big(T\big(\frac{k}{T} - K_r t_0\,(\bar{x}, n)\big)\big)$. Thus, in the discrete case, the pulse-compressed signal has the form

$$s_{pc}[n, t_k] \approx \int_{R_n} \sigma(\bar{x}) A_{rx}(\bar{x}, n) e^{-j2\pi f_0 t_0(\bar{x}, n)} \times T\,\mathrm{sinc}\left(T\left(\frac{t_k}{T} - K_r t_0\,(\bar{x}, n)\right)\right) dV \tag{2.26}$$

where the discrete variable t_k is used, as the convolution truly results in a time-domain signal. Here, there is a time-range mapping of $k = \frac{2K_r T \rho}{c} = \frac{\rho}{\Delta\rho}$.

2.4.2 FMCW Radar

For FMCW radars, most of the pulse compression has already been performed in hardware, but requires a Fourier transform to complete the process. Writing the Fourier transform of the intermediate signal in (2.14) as

$$s_{pc}\,(n, f) = \mathcal{F}\left\{s_{if}\,(n, t)\right\} \tag{2.27}$$

gives

$$\begin{aligned} s_{pc}\,(n, f) = \int_{-\infty}^{\infty} \frac{1}{2} \int_{R_n} & \sigma(\bar{x}) A_{rx}(\bar{x}, n, t) w_r\,(t - t_0\,(\bar{x}, n)) \\ & \times \cos\big(-2\pi\,(t_0\,(\bar{x}, n) - t_d)\,(f_0 + K_r t) \\ & + \pi K_r\left((t_0\,(\bar{x}, n))^2 - t_d^2\right)\big) dV \\ & \times e^{-j2\pi f t} dt. \end{aligned} \tag{2.28}$$

Solving the Fourier transform leads to the signal

$$\frac{1}{2}Te^{-j2\pi f t_0(\bar{x},n)}e^{\mp j2\pi K_r t_0(\bar{x},n)(t_0(\bar{x},n)-t_d)}e^{\mp j2\pi f_0(t_0(\bar{x},n)-t_d)} \times e^{\pm j\pi K_r\left(t_0(\bar{x},n)^2-t_d^2\right)}\operatorname{sinc}\left(T\left(f \pm K_r\left(t_0\left(\bar{x},n\right)-t_d\right)\right)\right). \tag{2.29}$$

These two sinc functions are images of each other. Here, the second term is chosen (where the $\pm$ in the sinc function is a minus sign) as this represents the positive frequency part. The pulse-compressed signal is then

$$\begin{aligned} s_{pc}\left(n,f\right) = \frac{1}{4}\int_{R_n}\sigma(\bar{x})A_{rx}(\bar{x},n,t)e^{j2\pi f_0(t_0(\bar{x},n)-t_d)}e^{-j\pi K_r\left(t_0(\bar{x},n)^2-t_d^2\right)} \\ \times e^{-j2\pi t_0(\bar{x},n)(f-K_r(t_0(\bar{x},n)-t_d))} \\ \times T\operatorname{sinc}\left(T\left(f-K_r\left(t_0\left(\bar{x},n\right)-t_d\right)\right)\right)dV. \end{aligned} \tag{2.30}$$

It can be seen in the above equation that there is a frequency-range mapping of $f = K_r\left(\frac{2\rho}{c}+t_d\right)$.

The result is slightly different in the case that the system transmits a down-chirp instead of the up-chirp used throughout this derivation. This can be represented by replacing K_r with $-K_r$. Due to this change, the first term of (2.29) is the positive frequency part, and so the resulting pulse-compressed signal has the form

$$\begin{aligned} s_{pc}\left(n,f\right) = \frac{1}{4}\int_{R_n}\sigma(\bar{x})A_{rx}(\bar{x},n,t)e^{-j2\pi f_0(t_0(\bar{x},n)-t_d)}e^{-j\pi K_r\left(t_0(\bar{x},n)^2-t_d^2\right)} \\ \times e^{-j2\pi t_0(\bar{x},n)(f-K_r(t_0(\bar{x},n)-t_d))} \\ \times T\operatorname{sinc}\left(T\left(f-K_r\left(t_0\left(\bar{x},n\right)-t_d\right)\right)\right)dV. \end{aligned} \tag{2.31}$$

Note the opposite sign of the f_0 exponential term, a result of the opposite chirp direction.

The recorded signal is, in practice, discrete, and with the Fourier transform there is the relation $f_k = \frac{k}{T}$ or $f_k = \frac{k-M}{T}$. Choosing the positive frequency-range relation during processing yields the discrete pulse-compressed signal

$$\begin{aligned} s_{pc}[n,k] = \frac{1}{4}\int_{R_n} \sigma(\bar{x}) A_{rx}(\bar{x},n,t) e^{j2\pi f_0(t_0(\bar{x},n)-t_d)} e^{-j\pi K_r\left(t_0(\bar{x},n)^2 - t_d^2\right)} \\ \times e^{-j2\pi t_0(\bar{x},n)(f - K_r(t_0(\bar{x},n)-t_d))} \\ \times T\,\mathrm{sinc}\left(T\left(\frac{k}{T} - K_r(t_0(\bar{x},n)-t_d)\right)\right) dV \end{aligned} \tag{2.32}$$

or

$$\begin{aligned} s_{pc}[n,k] = \frac{1}{4}\int_{R_n} \sigma(\bar{x}) A_{rx}(\bar{x},n,t) e^{-j2\pi f_0(t_0(\bar{x},n)-t_d)} e^{-j\pi K_r\left(t_0(\bar{x},n)^2 - t_d^2\right)} \\ \times e^{-j2\pi t_0(\bar{x},n)(f - K_r(t_0(\bar{x},n)-t_d))} \\ \times T\,\mathrm{sinc}\left(T\left(\frac{k}{T} - K_r(t_0(\bar{x},n)-t_d)\right)\right) dV \end{aligned} \tag{2.33}$$

for a down-chirp system. This results in the frequency-range mapping $k = \frac{2K_rT\rho}{c} + K_rTt_d = \frac{\rho}{\Delta\rho} + BWt_d$, which is the same as the discrete pulsed radar case if $t_d = 0$.

2.5 Summary

A model for a generic synthetic aperture radar system has been presented. This includes a description of how a SAR system takes advantage of platform motion in order to synthesize a large antenna array and achieve fine azimuth resolution. An introduction to the attitude orientation of the platform was given, to provide insight into its importance and to set up further discussion in later chapters. For both pulsed and FMCW radar systems, a detailed signal model is given, starting with the form of the transmitted pulse up through the recorded data. Additionally, the concept and signal model of pulse compression for both radar types is presented. This signal

model provides the basis for the formulation of the SAR processor presented here. The next chapter discusses various processing algorithms which attempt to properly form the SAR image by synthesizing the antenna array using the recorded data.

CHAPTER 3

PROCESSING ALGORITHMS

The signals recorded by a radar's data acquisition system will be discrete versions of the received signals

$$s_{if}[n, t_m] = s_{if}(n, mT_s). \tag{3.1}$$

Note that t_m is now a discrete time index, m, and that the brackets denote the entire signal as being discrete. From this point, it is the goal of the SAR processor to take these recorded signals and generate high resolution output imagery. The task is to generate a map (R_d) of backscatter values (σ) which amounts to inverting (2.5) or (2.14) for $\sigma(\bar{x})$. In this work, SAR processing algorithms are classified as either time-domain or frequency-domain algorithms based upon where they perform most of their essential processing steps.

3.1 Time-Domain Correlation

Among the methods for processing SAR data, perhaps the simplest and most obvious algorithm is the time-domain correlation algorithm. It is a two-dimensional matched-filter based on the recorded signal's impulse response [19]

$$\sigma(\bar{x}) = \sum_{n=1}^{N} \sum_{m=1}^{M} s_{if}[n, t_m]\, s_{\text{ref}}[n, t_m] \qquad \forall \bar{x} \in R_d. \tag{3.2}$$

In this formulation, the reference function takes the form

$$s_{\text{ref}}[n, t_m] = e^{j\phi_{\text{ref}}[n, t_m]} \tag{3.3}$$

where

$$\phi_{\text{ref}}[n, t_m] = 2\pi f_0 \frac{2\rho(\bar{x}, \bar{p}_{n,t_m})}{c} - \pi K_r \left(t_m - \frac{2\rho(\bar{x}, \bar{p}_{n,t_m})}{c} \right)^2 \tag{3.4}$$

for pulsed radars, and

$$s_{\text{ref}}[n, t_m] = \cos(\phi_{\text{ref}}[n, t_m]) \tag{3.5}$$

where

$$\begin{aligned} \phi_{\text{ref}}[n, t_m] =& 2\pi \left(\frac{2\rho(\bar{x}, \bar{p}_{n,t_m})}{c} - t_d \right) (f_0 + K_r t_m) \\ & - \pi K_r \left(\left(\frac{2\rho(\bar{x}, \bar{p}_{n,t_m})}{c} \right)^2 + t_d^2 \right) \end{aligned} \tag{3.6}$$

for FMCW radars.

In addition to the simplicity of this algorithm, it is also generally the most accurate due to the lack of simplifying assumptions which other algorithms (most notably the frequency-domain algorithms) make [19]. In this and all other algorithms, the accuracy is dependent on the precise knowledge of the motion of the radar platform. The main disadvantage of this algorithm, and the reason it is almost never used in practice, is that it requires processing time of order, $O(NM|R_d|)$ where N is the number of pulses, M is the number of time samples and $|R_d|$ is the size of the output image grid. Depending on the implementation, this algorithm could take approximately 10^4 times the real time data acquisition rate (so one minute of recorded data would take approximately one week to process).

3.2 Frequency-Domain Algorithms

One approach to improve the speed of processing is to perform some of the processing in the frequency-domain. Algorithms such as range-Doppler, ωK or chirp scaling, fall into this category [7]. These algorithms all begin with range/pulse compression, involving the *start-stop* approximation. From here, the range-Doppler algorithm performs the azimuth compression in the range-time, azimuth-frequency domain. The

general steps involve correcting the hyperbolic range migration for an assumed linear flight track

$$\rho = \sqrt{\rho_0^2 + v_0^2 t_{az}^2} \tag{3.7}$$

where v_0 is the nominal velocity and t_{az} is the slow time used to represent the distance traveled in the azimuth direction. This hyperbolic range is approximated as

$$\rho\left(f_{az}\right) \approx \rho_0 + \rho_0 \frac{\lambda^2 f_{az}^2}{8 v_0^2} \tag{3.8}$$

where f_{az} is the azimuth frequency. From this point, the azimuth portion of the matched filter can be applied as

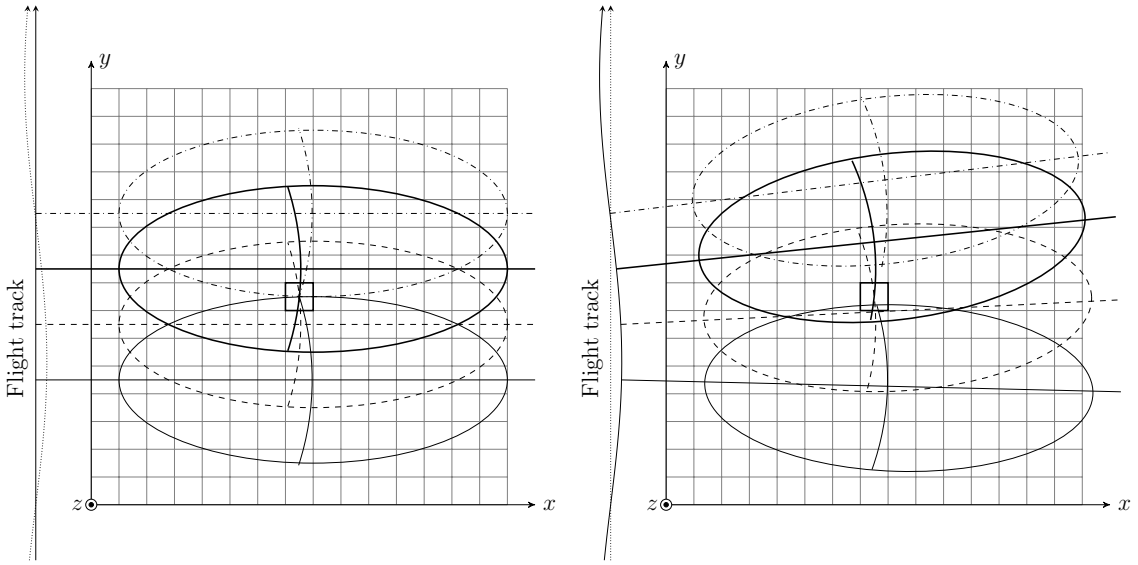
$$H_{az}\left(f_{az}\right) = e^{j\pi \frac{f_{az}^2 \lambda_c \rho_0}{2 v_0^2}}. \tag{3.9}$$

There are two major assumptions involved here. First is the assumption of a linear flight track used to define both the range migration and azimuth matched filter. Second is the approximation of the range migration hyperbola.

The ωK algorithm performs the azimuth compression in the range-frequency, azimuth-frequency domain, which avoids the approximation in the range migration correction of the range-Doppler algorithm, but still involves the assumption of a linear flight track.

Both of these frequency-domain algorithms have much faster computation times and can be quite accurate for some flight geometries. The ωK algorithm is also inherently good at processing scenes with high squint angles or large synthetic apertures. However, significant complexity, and non-trivial computation time, must be added to compensate for non-ideal platform motion.

A depiction of the effect of non-linear motion on the processing is shown in Figure 3.1. A generic frequency-domain processor will assume a linear flight track and



(a) Frequency-domain (b) Time-domain

Figure 3.1: Comparison of frequency and time-domain processing.

combine the returns from the beam footprints accordingly. In general, a time-domain processor will use the location of the beam footprints dependent on the non-linear motion of the platform. As is seen in the image, the non-linear motion will not only effect which beams illuminate the target, and where within the beam the illumination takes place, but also the range from the target to the platform. With such non-linear motion the range will no longer follow (3.7).

3.3 Time-Domain Backprojection

Another option is to apply the start-stop approximation, (2.16), to the time-domain correlation algorithm [16].

3.3.1 Pulsed Radar

Using the start-stop approximation, (3.4) becomes

$$\phi_{\text{ref}}[n, t_m] = 2\pi f_0 \frac{2\rho(\bar{x}, \bar{p}_{n,0})}{c} - \pi K_r \left(t_m - \frac{2\rho(\bar{x}, \bar{p}_{n,0})}{c} \right)^2. \tag{3.10}$$

This can be split into

$$\phi_{\text{ref}}\left[n, t_m\right] = \phi_{rn}\left[n\right]\phi_{rm}\left[n, t_m\right] \tag{3.11}$$

giving

$$\sigma\left(\bar{x}\right) = \sum_{n=1}^{N} e^{j\phi_{rn}[n]} \sum_{m=1}^{M} s_{if}\left[n, t_m\right] e^{j\phi_{rm}[n,t_m]} \qquad \forall \bar{x} \in R_d \tag{3.12}$$

where

$$\phi_{rn} = 2\pi f_0 \frac{2\rho(\bar{x}, \bar{p}_{n,0})}{c} \tag{3.13}$$

and

$$\phi_{rm} = -\pi K_r \left(t - \frac{2\rho(\bar{x}, \bar{p}_{n,0})}{c}\right)^2. \tag{3.14}$$

Noting that $(a-b)^2 = (b-a)^2$ and recalling that the definition of the pulsed compressed signal can be written in the discrete case as

$$s_{pc}[n,k] = \sum_{m=1}^{M} s_{if}[n,t_m] w_r\left(\frac{k}{K_r T} - t_m\right) e^{-j\pi K_r\left(\frac{k}{K_r T} - t_m\right)^2} \tag{3.15}$$

the inner sum of this expression can be replaced by

$$s_{pc}\left[n, K_r T \frac{2\rho(\bar{x}, \bar{p}_{n,0})}{c}\right] \tag{3.16}$$

yielding the backprojection algorithm

$$\sigma(\bar{x}) \approx \sum_{n=1}^{N} e^{j\phi_{\text{ref}}(\bar{x},n)} s_{pc}\left[n, \frac{\rho(\bar{x}, \bar{p}_{n,0})}{\Delta\rho} - \rho_{\text{ref}}\right] \qquad \forall \bar{x} \in R_d \tag{3.17}$$

which is approximate due to the start-stop approximation [16]. This uses the relation $\frac{2K_r T}{c} = \frac{2BW}{c} = \frac{1}{\Delta\rho}$. Note that an extra range-bias term, ρ_{ref}, is included for system specific calibration. In (3.17) the reference term has the form

$$\phi_{\text{ref}(\bar{x},n)} = 2\pi f_0 \frac{2\rho(\bar{x}, \bar{p}_{n,0})}{c}. \tag{3.18}$$

3.3.2 FMCW Radar

For FMCW radars, the reference phase term becomes

$$\begin{aligned}\phi_{\text{ref}}\left[n, t_m\right] =& 2\pi\left(\frac{2\rho(\bar{x}, \bar{p}_{n,0})}{c} - t_d\right)\left(f_0 + K_r t_m\right) \\ & - \pi K_r\left(\left(\frac{2\rho(\bar{x}, \bar{p}_{n,0})}{c}\right)^2 - t_d^2\right).\end{aligned} \tag{3.19}$$

Here, the reference function can be split using

$$\cos(\phi_{\text{ref}}\left[n, t_m\right]) = \frac{1}{2}\left(e^{j\phi_{\text{ref}}[n,t_m]} + e^{-j\phi_{\text{ref}}[n,t_m]}\right) \tag{3.20}$$

and again into

$$\phi_{\text{ref}}\left[n, t_m\right] = \phi_{rn}\left[n\right]\phi_{rm}\left[n, t_m\right] \tag{3.21}$$

giving

$$\begin{aligned}\sigma\left(\bar{x}\right) = \sum_{n=1}^{N}\frac{1}{2}\Bigg(& e^{-j\phi_{rn}[n]}\sum_{m=1}^{M} s_{if}[n, t_m]e^{-j\phi_{rm}[n,t_m]} \\ & + e^{j\phi_{rn}[n]}\sum_{m=1}^{M} s_{if}\left[n, t_m\right]e^{j\phi_{rm}[n,t_m]}\Bigg) \qquad \forall\bar{x} \in R_d\end{aligned} \tag{3.22}$$

where

$$\phi_{rn} = -2\pi f_0\left(\frac{2\rho(\bar{x}, \bar{p}_{n,0})}{c} - t_d\right) + \pi K_r\left(\left(\frac{2\rho(\bar{x}, \bar{p}_{n,0})}{c}\right)^2 - t_d^2\right) \tag{3.23}$$

and

$$\phi_{rm} = -2\pi K_r t_m\left(\frac{2\rho(\bar{x}, \bar{p}_{n,0})}{c} - t_d\right). \tag{3.24}$$

Again, given the definition of the pulsed-compressed signal in the discrete case

$$s_{pc}[n, k] = \sum_{m=1}^{M} s_{if}[n, t_m]e^{-j2\pi\frac{k}{T}t_m} \tag{3.25}$$

the inner sums can be replaced by the pulse-compressed signal as

$$s_{pc}\left[n, \mp TK_r\left(\frac{2\rho(\bar{x}, \bar{p}_{n,0})}{c} - t_d\right)\right] \tag{3.26}$$

where the second, positive-frequency term is kept, and the first term (the image), is thrown out during pulse compression. This gives the FMCW time-domain backprojection algorithm

$$\sigma\left(\bar{x}\right) = \sum_{n=1}^{N} \frac{1}{2} e^{j\phi_{ref}[n]} s_{pc}\left[n, \frac{\rho(\bar{x}, \bar{p}_{n,0})}{\Delta\rho} - BWt_d\right] \qquad \forall \bar{x} \in R_d. \tag{3.27}$$

In this case, the same time-domain backprojection algorithm is reached, as in (3.17), with a reference phase term of

$$\phi_{\text{ref}}(\bar{x}, n) = -j2\pi f_0\left(\frac{2\rho(\bar{x}, \bar{p}_{n,0})}{c} - t_d\right) + j\pi K_r\left(\left(\frac{2\rho(\bar{x}, \bar{p}_{n,0})}{c}\right)^2 - t_d^2\right) \tag{3.28}$$

and an extra factor of $\frac{1}{2}$ due to the single term.

The sum over the pulse time samples in the time-domain correlation algorithm has been replaced with a simple index into the range-compressed signal, and the overall algorithm is now just a sum over each pulse for each output pixel. This algorithm reduces the order of the computation time to $O(N|R_d|)$ which would be about 10^3 times faster than the time-domain correlation algorithm. While this may not be as fast as most frequency-domain algorithms, it is a large improvement over the time-domain correlation algorithm and only requires the start-stop approximation. This is usually seen as a favorable trade off between computation time and accuracy, as for most radars, the start-stop approximation is minor. The backprojection algorithm also will inherently correct for any non-linear aircraft motion unlike the frequency-domain algorithms.

3.4 Corrected Backprojection

For radar systems with long pulses and/or very large bandwidth, the start-stop assumption may introduce noticeable error. As described in the works of Ribalta and Stringham [16, 20], the majority of this error can be removed by using a first order approximation of the aircraft motion. More precisely, this is an approximation of the distance, ρ,

$$\rho(\bar{x}, \bar{p}_{n,t_m}) \approx \rho(\bar{x}, \bar{p}_{n,0}) + v_\rho[n]\, t_m \tag{3.29}$$

where

$$v_\rho[n] = |\bar{v}| \cos(\theta_{\bar{x},n}) \tag{3.30}$$

and

$$\theta_{\bar{x},n} = \arccos\left(\frac{\bar{v} \cdot (\bar{p}_{n,0} - \bar{x})}{|\bar{v}|\,|\bar{p}_{n,0} - \bar{x}|}\right). \tag{3.31}$$

The look angle, $\theta_{\bar{x},n}$, is the angle between the platform velocity vector, $\bar{v}$, and the look vector, $\bar{\rho}(\bar{x}, \bar{p}_{n,0})$. Note that the velocity, v_ρ, is one-dimensional and represents the rate of change of the distance to the target along the look vector, ρ, and not the platform velocity. This look vector velocity can then be written as

$$v_\rho[n] = \frac{\bar{v} \cdot (\bar{p}_{n,0} - \bar{x})}{|\bar{p}_{n,0} - \bar{x}|}. \tag{3.32}$$

3.4.1 FMCW Radar

The referenced derivations are for FMCW type radar systems. Therefore, this case is treated first. Using the linear approximation, (3.6) becomes

$$\begin{aligned}\phi_{\text{ref}}[n, t_m] =& 2\pi\left(\frac{2\rho(\bar{x}, \bar{p}_{n,0})}{c} + \frac{2v_\rho[n]t_m}{c} - t_d\right)(f_o + K_r t_m) \\ &- \pi K_r\left(\left(\frac{2\rho(\bar{x}, \bar{p}_{n,0})}{c} + \frac{2v_\rho[n]t_m}{c}\right)^2 - t_d^2\right).\end{aligned} \tag{3.33}$$

As in the uncorrected backprojection this may be expanded to

$$\sigma(\bar{x}) = \sum_{n=1}^{N} \frac{1}{2}\Bigg(e^{-j\phi_{rn}[n]} \sum_{m=1}^{M} s_{if}[n,t_m] e^{-j\phi_{rm}[n,t_m]} + e^{j\phi_{rn}[n]} \sum_{m=1}^{M} s_{if}\left[n,t_m\right] e^{j\phi_{rm}[n,t_m]} \Bigg) \qquad \forall \bar{x} \in R_d \tag{3.34}$$

where

$$\phi_{rn}[n] = -2\pi f_0 \left(\frac{2\rho(\bar{x},\bar{p}_{n,0})}{c} - t_d \right) + \pi K_r \left(\left(\frac{2\rho(\bar{x},\bar{p}_{n,0})}{c} \right)^2 - t_d^2 \right) \tag{3.35}$$

and

$$\begin{aligned} \phi_{rm}[n,t_m] =& t_m \left(-2\pi f_0 \left(\frac{2v_\rho\left[n\right]}{c} \right) - 2\pi K_r \left(\frac{2\rho(\bar{x},\bar{p}_{n,0})}{c} - t_d \right) \right) \\ &+ t_m \left(\pi K_r 2 \left(\frac{2v_\rho\left[n\right]}{c} \right) \left(\frac{2\rho(\bar{x},\bar{p}_{n,0})}{c} \right) \right) \\ &+ t_m^2 \left(-2\pi K_r \left(\frac{2v_\rho\left[n\right]}{c} \right) + \pi K_r \left(\frac{2v_\rho\left[n\right]}{c} \right)^2 \right). \end{aligned} \tag{3.36}$$

In this case, ϕ_{rn} has not changed. In order to reduce the second summation to a Fourier transform, as done above with (3.27), there can be no second order terms of the time component t_m in ϕ_{rm}. Ribalta notes that much of the correction is achieved with only the first term and uses the approximation [16]

$$\phi_{rm}[n,t_m] \approx t_m \left(-2\pi f_0 \left(\frac{2v_\rho\left[n\right]}{c} \right) - 2\pi K_r \left(\frac{2\rho(\bar{x},\bar{p}_{n,0})}{c} - t_d \right) \right). \tag{3.37}$$

Stringham takes the further approach of using a least squares approximation of t_m^2, given as [20]

$$t_m^2 \approx T t_m - \frac{T^2}{6}. \tag{3.38}$$

With this approximation, ϕ_{rn} becomes

$$\begin{aligned}\phi_{rn}[n] = &- 2\pi f_0\left(\frac{2\rho(\bar{x},\bar{p}_{n,0})}{c} - t_d\right) + \pi K_r\left(\left(\frac{2\rho(\bar{x},\bar{p}_{n,0})}{c}\right)^2 - t_d^2\right)\\ &+ 2\pi K_r\frac{T^2}{6}\left(\frac{2v_\rho\,[n]}{c}\right) - \pi K_r\frac{T^2}{6}\left(\frac{2v_\rho\,[n]}{c}\right)^2 \end{aligned} \tag{3.39}$$

and ϕ_{rm} becomes

$$\begin{aligned}\phi_{rm}[n,t_m] =&t_m\left(-2\pi f_0\left(\frac{2v_\rho\,[n]}{c}\right) - 2\pi K_r\left(\frac{2\rho(\bar{x},\bar{p}_{n,0})}{c} - t_d\right)\right)\\ &+ t_m\left(\pi K_r 2\left(\frac{2v_\rho\,[n]}{c}\right)\left(\frac{2\rho(\bar{x},\bar{p}_{n,0})}{c}\right)\right)\\ &+ Tt_m\left(-2\pi K_r\left(\frac{2v_\rho\,[n]}{c}\right) + \pi K_r\left(\frac{2v_\rho\,[n]}{c}\right)^2\right).\end{aligned} \tag{3.40}$$

Stringham notes that several further simplifications can be made. The second order velocity terms terms can be neglected as $v_\rho \ll c$. The v_ρ term in ϕ_{rn} is insignificant for practical radar pulse lengths and bandwidths. Finally the $v_\rho\rho(\bar{x},\bar{p}_{n,0})$ term in ϕ_{rm} is only significant when $\frac{2\rho(\bar{x},\bar{p}_{n,0})}{c} \sim T$ which is uncommon for FMCW radars.

With these approximations the phase terms reduce to

$$\phi_{rn}[n] = - 2\pi f_0\left(\frac{2\rho(\bar{x},\bar{p}_{n,0})}{c} - t_d\right) + \pi K_r\left(\left(\frac{2\rho(\bar{x},\bar{p}_{n,0})}{c}\right)^2 - t_d^2\right) \tag{3.41}$$

and

$$\begin{aligned}\phi_{rm}[n,t_m] = t_m\Bigg(&- 2\pi f_0\left(\frac{2v_\rho\,[n]}{c}\right) - 2\pi K_r\left(\frac{2\rho(\bar{x},\bar{p}_{n,0})}{c} - t_d\right)\\ &- 2\pi T K_r\left(\frac{2v_\rho\,[n]}{c}\right)\Bigg).\end{aligned} \tag{3.42}$$

The inner sums can now be replaced by the pulse-compressed signal (as done in the uncorrected case)

$$s_{pc}\left[n, T\left(\mp K_r\left(\frac{2\rho(\bar{x},\bar{p}_{n,0})}{c} - t_d\right) \mp f_0\frac{2v_\rho\,[n]}{c} \mp TK_r\frac{2v_\rho\,[n]}{c}\right)\right] \tag{3.43}$$

where the first, negative frequency, term is again ignored. Thus the corrected back-projection algorithm has the form

$$\sigma(\bar{x}) \approx \sum_{n=1}^{N} s_{pc}\left[n, \frac{\rho(\bar{x},\bar{p}_{n,0})}{\Delta\rho} - \rho_{\text{ref}} + \frac{f_0 v_\rho[n]}{K_r\Delta\rho} + \frac{T v_\rho[n]}{\Delta\rho}\right] e^{j\phi_{\text{ref}}(\bar{x},n)}$$

$$\forall \bar{x} \in R_d. \tag{3.44}$$

In this form, the third term in the range-compressed data's index is the correction term noted by Ribalta, and the fourth term is the additional correction used by Stringham. With this correction, the computational order is still $O(N|R_d|)$, like the standard backprojection algorithm, but does require the computation of the correction terms.

3.4.2 Pulsed Radar

The linear look velocity approximation in (3.29) may also be applied to pulsed radars, although with less usefulness due to the generally shorter pulse length. Here, reference phase term in (3.4) becomes

$$\begin{aligned}\phi_{\text{ref}}[n,t_m] =& 2\pi f_0\left(\frac{2\rho(\bar{x},\bar{p}_{n,0})}{c} + \frac{2v_\rho[n]t_m}{c}\right) \\ &- \pi K_r\left(t_m - \left(\frac{2\rho(\bar{x},\bar{p}_{n,0})}{c} + \frac{2v_\rho[n]t_m}{c}\right)\right)^2.\end{aligned} \tag{3.45}$$

Again, noting the form of the pulse-compressed signal

$$s_{pc}[n,k] = \sum_{m=1}^{M} s_{if}[n,t_m] w_r\left(\frac{k}{K_r T} - t_m\right) e^{-j\pi K_r\left(\frac{k}{K_r T} - t_m\right)^2} \tag{3.46}$$

the inner sum can be replaced given that the only phase term including t_m has the form $-\pi K_r\left(\frac{k}{K_r T} - t_m\right)^2$. The phase term is first expanded as

$$\begin{aligned}\phi_{\text{ref}}[n, t_m] =& 2\pi f_0 t_0 + 2\pi f_0 t_m t_v \\ & - \pi K_r \Bigg(t_m^2 - 2t_m t_0 - 2t_m^2 t_v \\ & \qquad + t_0^2 + 2t_m t_0 t_v + t_m^2 t_v^2 \Bigg)\end{aligned} \tag{3.47}$$

where $t_0 = \frac{2\rho(\bar{x}, \bar{p}_{n,0})}{c}$ and $t_v = \frac{2v_\rho[n]}{c}$ are rearranged using

$$1 - 2t_v + t_v^2 = (1 - t_v)^2 \tag{3.48}$$

to give

$$\begin{aligned}\phi_{\text{ref}}[n, t_m] =& 2\pi f_0 t_0 - \pi K_r (1 - t_v)^2 \Bigg\{ \\ & t_m^2 + t_m (1 - t_v)^{-2} \left(-2\frac{f_0}{K_r} t_v - 2t_0 (1 - t_v)^2 \right) \\ & + t_0^2 (1 - t_v)^{-2} \Bigg\}.\end{aligned} \tag{3.49}$$

Completing the square yields

$$\begin{aligned}\phi_{\text{ref}}[n, t_m] =& 2\pi f_0 t_0 - \pi K_r \Bigg\{ \\ & (1 - t_v)^2 \left[t_m + \frac{1}{2}(1 - t_v)^{-2} \left(-2\frac{f_0}{K_r} t_v - 2t_0 (1 - t_v)^2 \right) \right]^2 \\ & + t_0 - \frac{1}{4}(1 - t_v)^{-4} \left(-2\frac{f_0}{K_r} t_v - 2t_0 (1 - t_v)^2 \right)^2 \Bigg\}.\end{aligned} \tag{3.50}$$

To achieve the desired form, a small approximation is made

$$(1 - t_v) \approx 1 \tag{3.51}$$

which holds since $v_\rho[n] \ll c$. With this approximation, the phase term has the form

$$\begin{aligned}\phi_{\text{ref}}[n, t_m] =& 2\pi f_0 t_0 - \pi K_r \Bigg\{ \left[t_m + \frac{1}{2}\left(-2\frac{f_0}{K_r} t_v - 2t_0 \right) \right]^2 \\ &+ t_0^2 - \frac{1}{4}\left(-2\frac{f_0}{K_r} t_v - 2t_0 \right)^2 \Bigg\} \end{aligned} \tag{3.52}$$

and can be rearranged to give

$$\begin{aligned}\phi_{\text{ref}}[n, t_m] =& 2\pi f_0 t_0 - \pi K_r \left(t_0^2 - \left(\frac{f_0}{K_r} t_v + t_0 \right)^2 \right) \\ &- \pi K_r \left(t_m - \left(\frac{f_0}{K_r} t_v + t_0 \right) \right)^2 . \end{aligned} \tag{3.53}$$

Finally, being in the proper form, the inner sum can be replaced with

$$s_{pc}\left[n, K_r T \left(\frac{2\rho(\bar{x}, \bar{p}_{n,0})}{c} + \frac{f_0}{K_r} \frac{2v_\rho[n]}{c} \right) \right] \tag{3.54}$$

resulting in the algorithm form

$$\sigma(\bar{x}) \approx \sum_{n=1}^{N} s_{pc}\left[n, \frac{\rho(\bar{x}, \bar{p}_{n,0})}{\Delta\rho} - \rho_{\text{ref}} + \frac{f_0 v_\rho[n]}{K_r \Delta\rho} \right] e^{j\phi_{\text{ref}}(\bar{x}, n)} \quad \forall \bar{x} \in R_d \tag{3.55}$$

where

$$\begin{aligned}\phi_{\text{ref}}(\bar{x}, n) =& 2\pi f_0 \frac{2\rho(\bar{x}, \bar{p}_{n,0})}{c} \\ &+ \pi f_0 \frac{2v_\rho[n]}{c} \left(\frac{2\rho(\bar{x}, \bar{p}_{n,0})}{c} + \frac{f_0}{K_r} \frac{2v_\rho[n]}{c} \right). \end{aligned} \tag{3.56}$$

Note that the additional correction term to the pulse-compressed index is the same as the first term in the FMCW case. There is no second term derived here, but there is an additional term in the phase correction not present in the FMCW case.

In either pulsed or FMCW systems, the significance of these correction factors can be calculated. For the systems given in Table 5.1 the correction factors are calculated for various squint angles with the results shown in Table 3.1. As can be seen the

Table 3.1: Backprojection start-stop correction factors. The given correction factors can be compared to the approximate center swath range given. Note that the two terms shown for the FMCW systems represent the two correction terms.

	UAVSAR	UMass S-Band	UMass Ka-Band
Chirp Rate $\left[\frac{\text{GHz}}{\text{s}}\right]$	2500	100	100
Nominal Range [m]	15000	1500	1500
Correction Factor ($\theta_{sq} = 0°$) [m]	0.0058	0.18+0.0057	0.16+0.00048
Correction Factor ($\theta_{sq} = 10°$) [m]	0.025	0.49+0.015	3.5+0.010
Correction Factor ($\theta_{sq} = 20°$) [m]	0.043	0.77+0.024	6.7+0.019

correction factor is much more significant for the FMCW systems, especially the Ka-Band system with any noticeable squint. Also, since these systems have relatively short pulse lengths, the secondary correction factor is not as significant. For the pulsed system, the correction is small even for large squint angles. A pulsed system with a lower fractional bandwidth, $\frac{BW}{f_0}$, longer pulse length, T, or faster platform velocity would have a larger correction factor.

3.5 Summary

Starting from the recorded signal given in the signal model, several SAR processing algorithms have been presented. These algorithms attempt to generate the high resolution output imagery from the recorded data by taking advantage of the platform motion, adding coherently the returns from many pulses. Time-domain backprojection is presented as an alternative to the more common frequency-domain algorithms, due to its inherent ability to properly compensate non-linear platform motion. The form of time-domain backprojection is presented in detail, including correction factors designed to mitigate errors caused from the start-stop approximation. The next

chapter discusses SAR interferometry and how it may be different depending on the choice of SAR processing algorithm.

CHAPTER 4

BACKPROJECTION INTERFEROMETRY

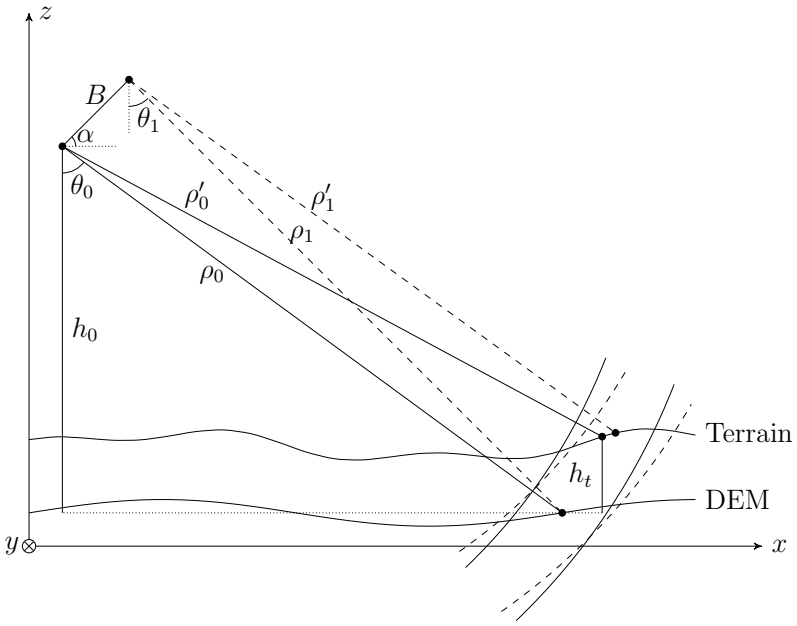


Figure 4.1: Interferometer geometry. The terrain here represents the position of the true or average phase center of the scatterers. ρ_0' and ρ_1' are the distances from the receive antennas to these points.

A standard SAR instrument is able to measure range and azimuth dimensions of scattering targets with precision. In the case that the targets lie on a known surface, their three-dimensional location can be given by the intersection of the range-azimuth dimensions and the surface. If the surface is not known precisely, other methods must be used. Most commonly this is achieved using interferometry [17]. A cross-track interferometer has two separate receiving antennas situated perpendicular to the direction of motion that receive simultaneously. Figure 4.1 shows this configuration, where the radar is flying in the y-direction, and the two antennas are separated by

a baseline distance, B, at angle, α. As is shown in the image, for a given range, the exact scatterer phase center will be different for each antenna, due to the difference in viewing geometry. Since the antennas are close together, the phase center displacement of the resolution cell will be very small and is usually ignored. For backprojection processing, the scatterer is assumed to lie on a DEM, as shown in Figure 4.1. In some cases, the range to the scatterer assumed position, ρ, and its actual position, ρ', will be the same. However, non-linear platform motion and phase center deviations from the pixel center will complicate the issue, and it may not be possible to assume $\rho = \rho'$. A simplification of the general case to two dimensions, as shown, is presented here.

Part of interferometric processing is estimating the complex coherence, $\bar{\gamma} = |\bar{\gamma}|e^{j\phi}$, whose phase represents the interferometric phase. The coherence can be estimated from the imagery as [22]

$$\bar{\gamma} = \frac{\sum_{i=1}^{L} z_{0i} z_{1i}^*}{\sqrt{\sum_{i=1}^{L} |z_{0i}|^2} \sqrt{\sum_{i=1}^{L} |z_{1i}|^2}} \tag{4.1}$$

where z_0 is the first channel, z_1 is the second channel and L is the number of looks used. In two dimensions, this has the form

$$\bar{\gamma} = \frac{\sum_{i=1}^{I} \sum_{j=1}^{J} z_0[i,j] z_1^*[i,j]}{\sqrt{\sum_{i=1}^{I} \sum_{j=1}^{J} |z_0[i,j]|^2} \sqrt{\sum_{i=1}^{I} \sum_{j=1}^{J} |z_1[i,j]|^2}} \tag{4.2}$$

where I is the number of looks in the azimuth direction and J is the number of looks in the range direction, leading to $L = IJ$ total looks.

4.1 Traditional Interferometry

Traditional interferometry aims to use the residual processed phase to precisely measure the incidence angle of the scatterer, θ_0'. The major assumption made in this

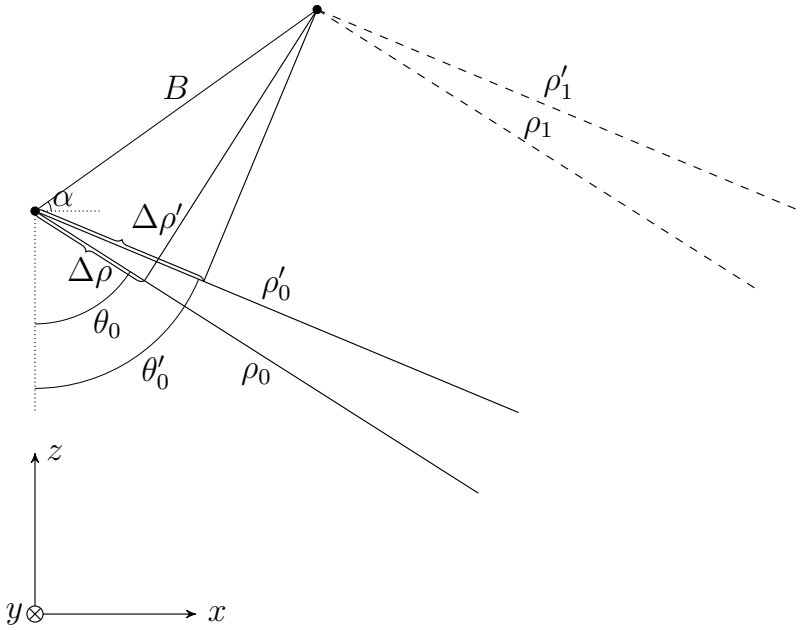


Figure 4.2: Traditional interferometry detail.

case is that the look vectors $\bar{\rho}_0'$ and $\bar{\rho}_1'$ are parallel [17]. This assumption derives from the fact that $B \ll \rho$ (where ρ may be any of the involved ranges). By making this assumption, the difference in length between the look vectors $(\rho_0' - \rho_1')$ is simply one side of the right triangle shown in Figure 4.2, labelled as $\Delta\rho'$. Solving this triangle for $\Delta\rho'$ yields

$$\Delta\rho' = B\sin(\theta_0' - \alpha) \tag{4.3}$$

thus

$$\theta_0' = \alpha + \arcsin\left(\frac{\Delta\rho'}{B}\right). \tag{4.4}$$

Using the general interferometer geometry shown above, this may be converted to a height as

$$h_t = h_0 - \rho_0'\cos\left(\alpha + \arcsin\left(\frac{\Delta\rho'}{B}\right)\right). \tag{4.5}$$

This approach is derived with frequency-domain processing in mind. With frequency-domain processing, the output grid is truly processed in the slant range, meaning the

image ranges are ρ' values. Additionally the image phase for each antenna depends on the path propagation length, and will have the form

$$\psi = -\frac{a2\pi}{\lambda}\rho' \tag{4.6}$$

where $a = 1$ for bistatic or "single antenna transmit" (SAT) operation, and $a = 2$ for monostatic or "ping-pong" operation [17]. For SAT operation one of the antennas transmits and both receive, while for ping-pong operation, each antenna transmits and receives separately. In this case the interferometric phase, which is defined as the phase difference between the two data sets, has the form

$$\phi = \psi_0 - \psi_1 = \frac{a2\pi}{\lambda}(\rho'_1 - \rho'_0). \tag{4.7}$$

Using the traditional interferometry derivation, namely the approximation $\Delta\rho' \approx \rho'_0 - \rho'_1$ the phase may also be written as [17]

$$\phi = -\frac{a2\pi}{\lambda}B\sin(\theta'_0 - \alpha) \tag{4.8}$$

Thus, the interferometric phase can be used to estimate the height of the scatterer

$$h_t = h_p - \rho_0 \cos\left(\alpha + \arcsin\left(-\frac{\phi\lambda}{a2\pi B}\right)\right). \tag{4.9}$$

4.2 Backprojection Interferometry

Duersch develops another approach to interferometry based on backprojection processing [8]. This development starts with the geometry and imagery of a back-projection processed scene, but for the generic case shown in Figure 4.1, the same result may also be derived from the two-dimensional geometry. This involves two major assumptions (which are also made eventually in Duersch's approach). First,

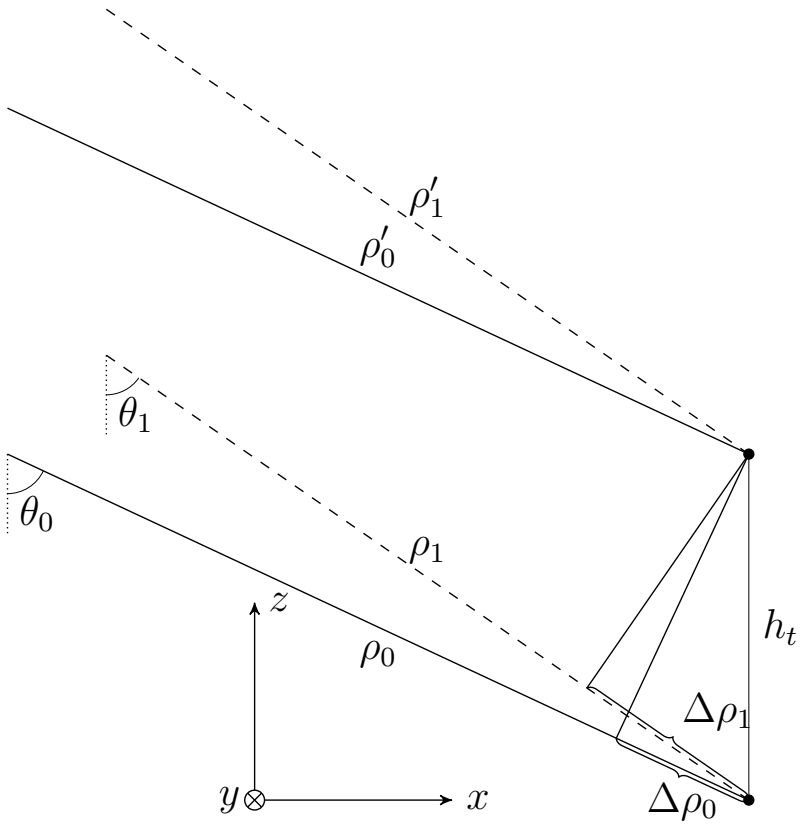


Figure 4.3: Backprojection interferometry detail.

is that the displacement of the scatterer from the reference point is purely in the vertical direction. This will hold for $\theta \gg 0$. Second, is that the look vectors ρ_0 and ρ_0' are parallel, and the vectors ρ_1 and ρ_1' are parallel. This is derived from the assumption that $h_t \ll \rho$. These assumptions lead to the scatterer configuration shown in Figure 4.3. Similar to the traditional case, right triangles may be constructed to determine path length differences, however these path lengths are instead

$$\Delta\rho_0 \approx \rho_0 - \rho_0' \tag{4.10}$$

and

$$\Delta\rho_1 \approx \rho_1 - \rho_1'. \tag{4.11}$$

Here, the height may be directly determined by

$$h_t = \frac{\Delta\rho_0}{\cos(\theta_0)} = \frac{\Delta\rho_1}{\cos(\theta_1)} \tag{4.12}$$

In backprojection, the received phase still has the same form as (4.6), but the range dependence is mostly removed during processing. Instead, the phase for each antenna will have the form

$$\psi = \frac{a2\pi}{\lambda}(\rho - \rho') \tag{4.13}$$

where $(\rho - \rho')$ is the residual or uncorrected range due to a displacement of the scatterer [8]. This is illustrated in Figure 4.1 where the target has been assumed by the processor to lie upon the DEM, but the true phase center of the scatterer measured by the antennas is in another location. In this case the interferometric phase will be

$$\phi = \psi_0 - \psi_1 = \frac{a2\pi}{\lambda}((\rho_0 - \rho_0') - (\rho_1 - \rho_1')). \tag{4.14}$$

Using the approximations of the backprojection interferometry approach, the phase can be written as

$$\phi = \frac{a2\pi}{\lambda}(\Delta\rho_0 - \Delta\rho_1) = \frac{a2\pi}{\lambda}(h_t\cos(\theta_0) - h_t\cos(\theta_1)) \tag{4.15}$$

and inverted to solve for height as

$$h_t = \frac{\phi}{\frac{4\pi}{\lambda}(\cos\theta_0 - \cos\theta_1)} \tag{4.16}$$

which is the same result as derived by Duersch [8]. Note that there are significant differences from the height estimate used with traditional interferometry.

Duersch shows a detailed comparison of sensitivities for traditional versus backprojection interferometry approaches [8]. To do so, (4.16) is rewritten in terms of the traditional parameters by using the law of sines with the original $(\rho_0\rho_1 B)$ triangle

$$h_t = \frac{\phi}{\frac{4\pi}{\lambda}\left(\cos\theta_0 - \cos\left(\theta_0 - \arcsin\left(\frac{B}{\rho_1}\cos(\alpha - \theta_0)\right)\right)\right)}. \tag{4.17}$$

Several important differences are found. Traditional interferometry is less sensitive to the interferometric phase. Backprojection interferometry is shown to be less sensitive to geometric errors, notably the baseline length and angle. Unlike traditional interferometry, backprojection interferometry becomes less sensitive to errors as the baseline length increases. Also, with backprojection interferometry, the interferometric phase should vary slowly as a function of height displacement or incidence angle. From the differences found it is expected that backprojection interferometry would perform better in some cases, most notably with large baselines or platform geometry measurement errors.

It should be noted that while the "backprojection interferometry" approach may be derived from the backprojection processed imagery and may be convenient to

use for such images, the traditional approach may also be used with backprojection images[1]. To do so, the additional phase added to each channel during the processing, namely $\frac{a2\pi}{\lambda}\rho$, could also be removed during the processing, leading to the same phase form as given in the traditional case. At this point, the traditional method could be employed as usual, especially easily if, like is done in the processor presented here, the output is in a slant range grid. Another option would be to remove the additional phase after processing, giving the result

$$\theta_0' = \alpha + \arcsin\left(\frac{\rho_1 - \rho_0}{B} - \frac{\phi\lambda}{a2\pi B}\right). \tag{4.18}$$

Using an assumption similar to that used in the traditional approach, $\rho_1 - \rho_0 \approx \Delta\rho$, as seen in Figure 4.2, this could also be written as

$$\theta_0' = \alpha + \arcsin\left(\sin(\theta_0 - \alpha) - \frac{\phi\lambda}{a2\pi B}\right). \tag{4.19}$$

4.3 Summary

Two different approaches to cross-track interferometry are presented; the traditionally used approach derived from frequency-domain processing and a new approach derived from time-domain backprojection. The new approach is well suited to backprojection processing, due to the nature of the processor's outputs, and is shown to perform differently than the traditional approach. The performance difference should give advantages to each approach for certain radar system configurations. It is noted that, if care is taken, either interferometric approach could be used with backprojection processed imagery. Having developed a suitable signal model and processing algorithm, the next chapter will discuss general details which must be considered

[1] In fact the "backprojection interferometry" approach could also likely be used with slant-range, frequency-domain processed imagery.

in implementing the time-domain backprojection algorithm, and provides a detailed look into the implementation created as a part of this thesis.

CHAPTER 5

IMPLEMENTATION DETAILS

This thesis presents details of the SAR Processor under development at UMass. It has the goal of creating highly accurate focused imagery from both the UAVSAR platform developed and operated by NASA's Jet Propulsion Laboratory as well as two radar platforms under development at UMass Amherst. These two platforms are both side-looking cross-track interferometers. One operates at Ka-band using slotted waveguide antennas and the other at S-band using microstrip patch antennas. The important specifications of these systems are listed in Table 5.1.

Table 5.1: Parameters of the relevant radar systems.

	UAVSAR	UMass S-Band	UMass Ka-Band
Center Frequency [GHz]	1.2575	3.2	34.945
Pulse Length [μs]	40	1000	1000
PRF [Hz]	500	1000	1000
Beam Width [°]	6	12	1
Synthetic Aperture Length [m]	2200	350	30
Altitude [km]	12.5	0.5-2	0.5-2
Velocity [m/s]	220	55	55
Baseline [m]	Variable	0.756	0.1

The parameters in Table 5.1 show the systems for which the processor is to be used have a great deal of variation. There is a wide range of center frequencies and beam widths as well as platform speeds and altitudes. Additionally, the UMass radars are both FMCW, while UAVSAR is a pulsed system. Due to these varying parameters, the processor must be flexible enough to work well with many types of input data.

Because of the large non-linear motion of the low altitude radars, and the desire to achieve greater accuracy with airborne platforms, the processor is designed around

the backprojection algorithm. In order to operate at a reasonable speed, and to be relatively easy to write and maintain, the processor is being written using the CUDA language to perform the core focusing tasks, and with Python to perform all of the host functions. This hybrid approach allows the processor to take advantage of the speed benefits of GPU hardware in regards to backprojection without becoming too complex and difficult to maintain.

5.1 GPU Hardware

While the backprojection algorithm is a large improvement in speed over the time-domain correlation algorithm, it still has trouble keeping up with frequency-domain based approaches. A unique feature of the algorithm developed in this thesis is that it performs almost exactly the same computation for each output pixel. In fact, a trivial implementation would include a loop over every pixel which performs the sum in (3.17). This type of implementation is an excellent candidate for running in parallel. Given that the algorithm is nearly the same for every pixel, is fairly simple, and the fact that there will be a very large number of pixels, it is a good candidate to be run on a GPU.

Graphics processing units (GPUs) have become a useful computational tool for parallel processing. Modern GPUs have thousands of processing cores, coupled with gigabytes of onboard memory, and are designed with general purpose parallel computing in mind. This makes them a very good candidate for SAR processing with backprojection.

The basic unit of the NVIDIA GPU architecture is known as the streaming multiprocessor (SMP) [6]. It is a collection of processing cores and registers, instruction and scheduling, cached and shared memory and separate processing units for double precision and transcendental functions. The exact layout of these units within the SMP changes between generations of NVIDIA GPUs. At a higher level, the

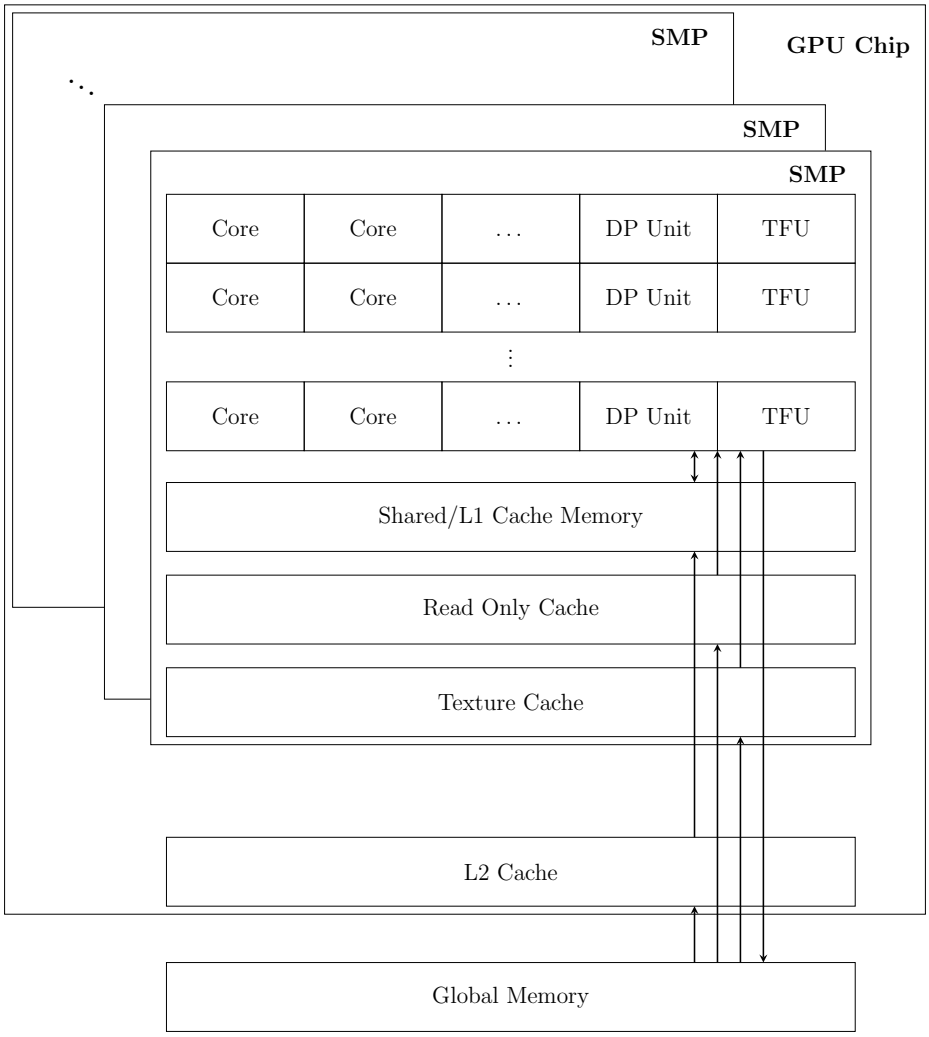


Figure 5.1: GPU hardware layout.

GPU contains many streaming multiprocessors as well as additional cache memory, overall thread scheduling and external memory controllers. GPU cards will contain large amounts of memory (on the order of gigabytes), separate from the GPU chip itself, which is accessed by the GPU as global memory. This memory can be directly accessed by the host CPU.

The SMP will schedule blocks of 32 computational cores, called warps, to execute the same instructions in what NVIDIA calls single instruction multiple thread (SIMT) commands. This is distinct from single instruction multiple data (SIMD) commands in that it allows each core to branch independently. It should be noted that while cores are allowed to branch independently, such independent branches are executed serially, and for this reason, branching will have a significant impact on performance. Individual operating cores have access to their local register file and the shared memory on the SMP. The shared memory allows cores to communicate and access the same memory space without going through the entire memory cache. Cores can also read and write to global memory through the layers of on board memory caches. There are several ways that the global memory can be laid out and accessed by the processing cores which can improve the memory performance. There is a small amount of constant memory which allows cores within a warp to collect memory instructions into a single memory access. If the cores are all accessing the same memory address, this allows significant improvement, but will be slower than generic global memory if they are not. The other type of memory is known as texture memory. The texture memory is cached in such a way as to allow improved memory access speed when cores request memory at spatially (in one, two or three dimensional grids) nearby locations. The GPU is also capable of performing hardware based linear interpolation of texture memory, allowing cores to use floating point indexing.

At the time of this writing, a modern NVIDIA GPU (the Tesla K40 for example) has 192 single-precision cores, 64 double-precision units and 32 transcendental func-

tion units per streaming multiprocessor [14]. Each SMP also has 64KB split between shared memory and the L1 cache as well as 64KB for read-only memory and the texture cache. The register file on each SMP contains 65536 32-bit registers. The Tesla K40 contains 15 SMP units for a total of 2880 processing cores and has 12GB of onboard global memory.

For general purpose computing on their GPUs, NVIDIA provides the CUDA programming language [6]. CUDA is an extension to the C programming language designed to enable straightforward processing of their SIMT architecture as well as a set of drivers and tools to run, debug and benchmark CUDA programs. Programming with CUDA involves writing a kernel program, which will be executed on the GPU as well as a host program. The host program will run on the host computer's CPU and will set up, run and retrieve the results of the CUDA kernel.

In the CUDA language, there is a layout which closely matches the hardware layout. The CUDA kernel is the SIMT program which will be run on every GPU core used. Here, every instance of the kernel is known as a thread, which is basically synonymous with a GPU core. The threads are arranged into one-, two- or three-dimensional groups called blocks which are further grouped into one-, two- or three-dimensional groups called grids. Blocks are then relatable to streaming multiprocessors, and grids to groups of SMPs, but this is a loose relation. There is a limit to the number of threads per block, and blocks per grid, but the total number of threads per block can exceed the size of the SMP, and the total number of threads in the GPU program can also exceed the total number of processing cores on the GPU. This is to allow the thread-, block- and grid-layout of a particular program to be based on the dimensionality of the problem and not be constrained by the hardware specifics. Though the kernel program is the same for each thread, when it is run, the thread has access to its index within its block, and the index of its block within the grid. This allows the separate threads to perform the correct operations based

on how the host has set up the block and grid layout. When executing, blocks are broken into groups of 32 threads, known as warps, by the scheduler. It is guaranteed that these warps contain 32 consecutive threads from the block, which is necessary to allow the correct and efficient use of the shared memory and other memory caches. Beyond this, the order in which separate warps are executed, and the order of execution of separate blocks, is not guaranteed. Thus blocks must be independent, and additionally, all blocks must be able to run in parallel, as the blocks that are chosen to run at any given time is up to the GPU scheduler. This general functionality of the CUDA language will dictate how a kernel is written and run, for any given problem.

5.2 Algorithm Details

The general layout of the processor is as follows. First, all of the ephemeris information must be compiled and a single structure representing the antenna channel of interest is created. This structure represents not only the platform position, $\bar{p}_{n,0}$, but also the antenna attitude used to calculate the antenna squint angle. The importance of the antenna squint angle will be shown later on. Once this structure exists, the processor can perform the range compression, and then proceed to execute the backprojection algorithm given in (3.17) on the GPU. The layout of the processor is depicted in Figure 5.2.

In order to perform the backprojection algorithm, the processor must have, or be able to calculate, certain information. Various parameters of the radar must be known in order to calculate the correct constants used throughout the algorithm. The range-compressed data must be available which is easily calculated ahead of time. The positions $\bar{x}$ and $\bar{p}_{n,0}$ must be known (in full three dimensions of the same coordinate system) in order to calculate the range to target, ρ. The positions, $\bar{x}$, which represent locations on the ground, are generated from a user defined grid, R_d, and possibly use a digital elevation map (DEM) when translating this grid to 3D ground coordinates.

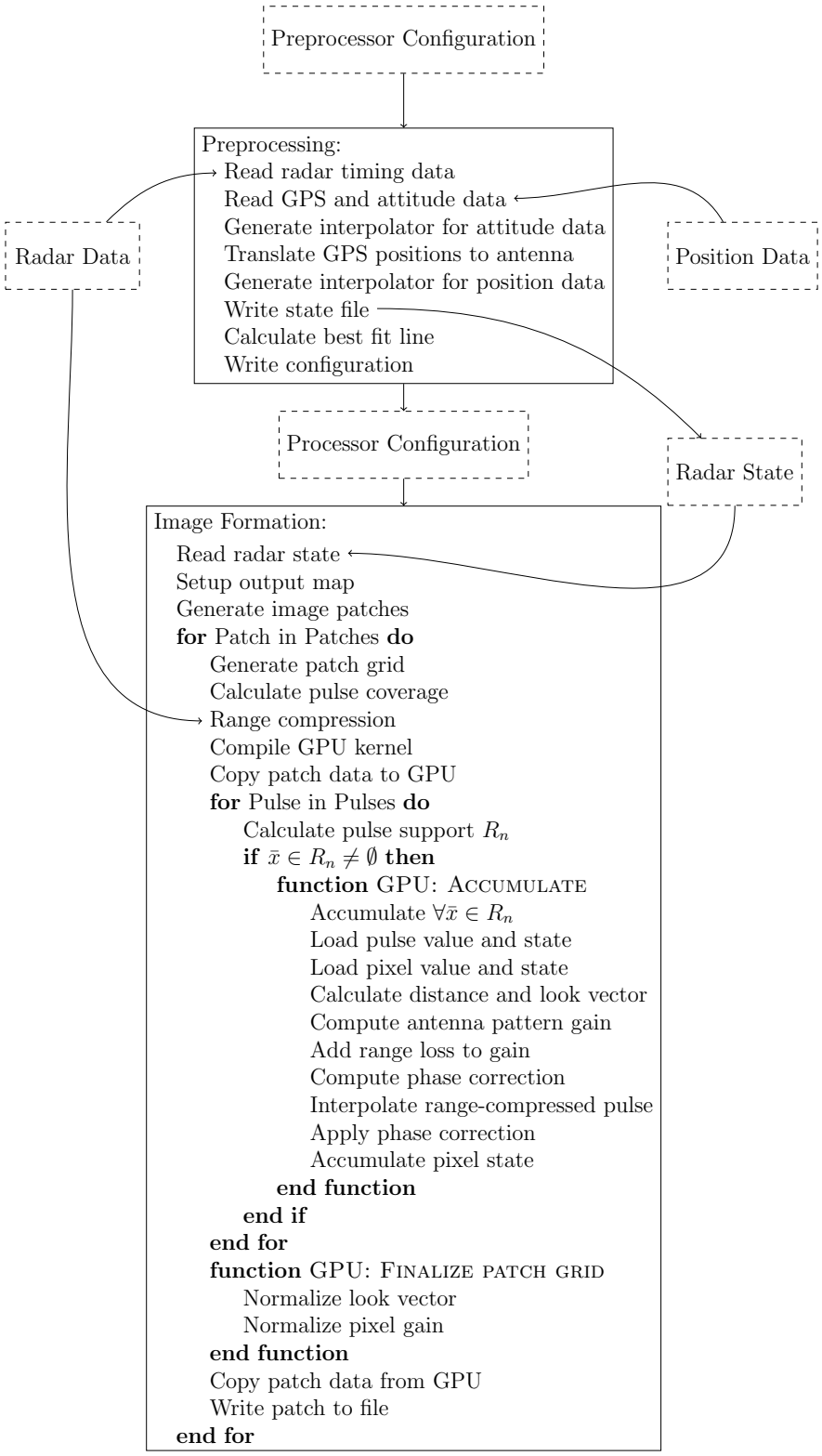


Figure 5.2: Processor block diagram/pseudocode. Dashed boxes represent files on disk.

The positions, $\bar{p}_{n,0}$, which represent the locations of the platform, are determined from the platform ephemeris structure. If using the corrected backprojection algorithm, these vectors will also be used to calculate the look vector speed, v_ρ.

An important aspect which has not yet been discussed in detail is the first part of the algorithm, which involves the sum over all radar pulses, $\sum_{n=1}^{N}$. While the processor would work if every available pulse, n, were summed for every output pixel, $\bar{x}$, there are several reasons why this would lead to errors in the processing. This stems from the fact that the antenna has a finite azimuth beamwidth. Though the antennas will radiate to, and receive power from, nearly all directions, the majority will be concentrated in the main lobe of the antenna beam, which has a finite size. Despite the fact that power from the side lobes will almost always be lower than the power received from the main lobe, and often lower than the noise floor of the system, the nature of the coherent phase-corrected sum that the algorithm performs will allow some of this power to contribute constructively to output image pixels. Since these radars nominally operate in stripmap mode (i.e. creating a range and azimuth swath as the platform travels along an ideally linear flight track), for any given scene there will be far more pulses whose side lobes illuminate any given pixel than pulses whose main lobe illuminates that pixel. Given the fact that this side lobe power will almost always be far below the peak recorded power (especially in the far lobes), a lot of noise will be added into each output pixel. [1] Additionally, the effect of errors in the platform ephemeris are enhanced at further ranges which would exist for pulses far from the target pixel. Thus, the output image quality will be most likely be degraded, rather than enhanced, by including all of the pulses. Another reason to avoid this technique is that it requires much more computation compared to using a smaller set of pulses.

[1] Here noise can also refer to the recorded return of a target which is not located at the desired focusing location.

For these reasons, it is desirable to replace the sum over all pulses with the set of pulses where $\bar{x} \in R_n$, which is denoted as $\sum_{\{n:R_n \ni \bar{x}\}}$. In other words, the value of the reflectivity at pixel $\bar{x}$ should be the sum of the phase-corrected (and properly indexed) range-compressed values for every pulse which illuminates the pixel within its main beam. Using this definition of the sum requires calculating the beam footprint R_n. Fortunately, for stripmap SARs, this is not difficult to do given some minor assumptions. The calculation does require knowledge of the platform position and attitude for each pulse, both of which can be derived from the state structure.

Additionally, the amplitude term, $A_{rx}(\bar{x}; n)$, can be corrected. This term contains the gain due to the antenna pattern, the radiometric gain of the radar hardware and any other propagation losses. The propagation losses will depend mostly on the range, and can be removed from the range-compressed data prior to image formation. The radiometric gain will ideally be slowly varying with time and frequency independent, and can also be removed ahead of time. The antenna pattern will depend on the antenna position, attitude and pixel location, and so must be removed during image formation. This gain correction will also correct for the varying sizes of the set $\{n : R_n \ni \bar{x}\}$ and enables proper estimates of the reflectivity, σ^0.

5.2.1 Non-Equispaced Fast Fourier Transform

Another important aspect of the algorithm is the indexing of the range-compressed data, which is given as

$$s_{pc}\left[n, \frac{\rho(\bar{x}, \bar{p}_{n,0})}{\Delta\rho} - \rho_{\text{ref}}\right] \tag{5.1}$$

for the uncorrected backprojection. What is important to note is that the range-compressed data, s_{pc}, is discrete, while the index term depends on the continuous variable, ρ. Since the phase correction applied during image formation is a very sensitive function of the range, the accuracy of the processor will depend on how the range-compressed data are interpolated. For this step, nearest neighbor interpola-

tion will not suffice. Several methods of interpolation have been discussed and used in practice including nearest neighbor, linear, cubic and Knab interpolation, all of which can be accompanied by an oversampling of the range-compressed data [3]. Another method which has more recently been developed, and used for radar processing, is known as the Non-Equispaced Fast Fourier Transform (NERFFT) [9] which is applicable due to the fact that the range-compressed data for both pulsed, and FMCW radars, are computed using FFTs.

Ideally the processor works by sampling the forward or inverse Fourier transform of the signal $x(k)$ (s_{if} in the FMCW case) at non-uniform locations, l, given by the range-dependent index term. These can be calculated exactly, simply by employing the definition of the Fourier transform for the given point of interest

$$X(l) = \sum_{k=-N/2}^{N/2-1} e^{-j2\pi lk/N} x(k). \tag{5.2}$$

This would avoid the need for the FFT, but require a sum over all of the data samples, resulting in an algorithm of the same computational order as the time-domain correlation algorithm. The values can also be calculated exactly from the uniformly Fourier transformed (range-compressed) signal using ideal, or Dirichlet, interpolation but again this requires a sum over all samples.

In his derivation of the NERFFT, Fourmont [9] shows that evaluation of a Fourier transform of a signal at non-uniform points can be evaluated as

$$X(l) = \frac{1}{\sqrt{2\pi}} \sum_{m \in \boldsymbol{Z}} \hat{\phi}(cl - m) \sum_{k=-\gamma N/2}^{\gamma N/2-1} e^{-j2\pi mk/(\gamma N)} \frac{x(k)}{\phi(2\pi k/(\gamma N))} \tag{5.3}$$

with some minor assumptions. Here, ϕ is a window function and $\hat{\phi}$ is its Fourier transform. The factor, γ, is an oversampling factor. The window, ϕ, is required to be non-zero over the interval $\left[-\frac{\pi}{\gamma}, \frac{\pi}{\gamma}\right]$, piecewise continuously differentiable over $[-\alpha, \alpha]$

and vanishing outside $[-\alpha, \alpha]$. The parameter, α, is defined such that $0 < \frac{\pi}{\gamma} < \alpha$ and $\alpha < \pi\left(2 - \frac{1}{\gamma}\right)$. What is important to note is that the inner sum does not contain l, and is just an oversampled and windowed version of the DFT which can be computed with a standard FFT. Also, given the above definitions, a window function can be chosen whose value is small outside a small interval in m, such that the outer summation can be reduced to a sum from $-K$ to K. A window function which is tractable and approximates this behavior, is the Kaiser-Bessel window where

$$\phi(\zeta) = \begin{cases} I_0\left(K\sqrt{\alpha^2 - \zeta^2}\right) & |\zeta| \leq \alpha \\ 0 & |\zeta| > \alpha \end{cases} \tag{5.4}$$

and

$$\hat{\phi}(\psi) = \sqrt{\frac{2}{\pi}} \frac{\sinh\left(\alpha\sqrt{K^2 - \psi^2}\right)}{\sqrt{K^2 - \psi^2}}. \tag{5.5}$$

Since the window function can be precomputed, the core of this interpolation method involves computing a windowed, oversampled FFT as part of the range compression, and performing a sum over a small interval $[-K, K]$ to interpolate the desired value.

The accuracy of this method depends on the chosen window function and the values of the parameters γ and K, and is shown to be far more accurate than other interpolation methods for similar computational cost. It has been shown that the error in the interpolation reaches the level of single precision accuracy at $\gamma = 2$ and $K = 3$ and double precision accuracy at $\gamma = 2$ and $K = 6$ [9].

5.3 Motion Estimation and Autofocus

Since the backprojection processor requires precise knowledge of the antenna position and attitude, effort must be taken to ensure that they are measured accurately. Generally, these are measured using a combination of a GPS unit for position information, and an inertial navigation unit (INU) for attitude information. The information

from these sensors must also be used alongside knowledge of the spatial location of the sensors relative to the radar antenna. When referring to the antenna, this is the location of the antenna phase center and the orientation of the beam center, both of which would usually be measured in an antenna chamber. With this knowledge, measurements (perhaps better stated as estimates) of the antenna position and attitude, can be made for each radar pulse. The accuracy of these estimates will depend on the accuracy of the sensors themselves, the measurements of the spatial offsets and, additionally, the proper alignment in time of the sensor information and the radar data.

The quality of the imagery produced by the processor will depend on the accuracy of these estimates, and can be severely degraded if they are of poor quality. One way to improve the quality of these estimates is to use properties of the recorded radar data to measure the antenna motion.

Doppler estimation is a common technique used to measure the Doppler frequency of recorded radar data. The Doppler frequency, which is a measure of the frequency content of the radar data in the azimuth direction, can be used as a proxy for the radar squint angle. Further, the squint angle, as seen in Figure 2.2, is a function of the antenna yaw and pitch attitude angles. The squint angle can be given as

$$\theta_{sq} = \arctan\left(\frac{\cos(\theta_y)\sin(\theta_p)\frac{h}{\cos(\theta_p)}(1-\tan^2(\theta_y)) - (\text{dir})\tan(\theta_y)\sqrt{\rho_0^2 - h^2}}{\rho_0}\right) \tag{5.6}$$

where (dir) is either +1 for left looking radars or -1 for right looking radars, and ρ_0 is the radar slant range. Additionally, the Doppler frequency can be shown to be [7]

$$f_{az} = \frac{2v_{az}\sin(\theta_{sq})}{\lambda}. \tag{5.7}$$

One way of measuring the Doppler frequency from recorded radar data is to use the phase of the range-compressed image. It holds that

$$f_{az}[n,r] \approx \frac{\angle\{s_{pc}[n,r]s^*_{pc}[n-1,r]\}}{2\pi\Delta T} \tag{5.8}$$

where $\angle\{x\}$ is the complex phase of x and ΔT is the pulse repetition time defined as $\Delta T = \frac{1}{\text{PRF}}$ [7]. In practice, this estimate of f_{az} will be averaged in both range and azimuth directions to reduce noise in the measurement. Using this estimate, a fitting procedure can be performed to provide an estimate of the parameters of the Doppler model, most notably the yaw and pitch angles.

The magnitude of the range-compressed image can also be used as an estimate for the Doppler frequency. Additionally, much work can be done to estimate the quality of the Doppler estimation at any given point in order to improve the fitting procedure [7]. A simple method is just to note that the expected SNR will decrease with increasing range and weight the fits accordingly.

Autofocus is another method to correct for unknown, or uncorrected, platform motion (as well as other phase errors) that has been commonly applied to frequency-domain algorithms [7]. The basic idea is to apply a phase correction during image formation, which is based upon maximizing the image quality directly and generally bypassing any calculations of the source of the phase errors (as in direct motion estimation). How the correction is calculated, and applied, varies between different methods. Similar methods could potentially be applied to a backprojection processor in addition to, or instead of, estimating the unknown platform motion.

The initial results from the different studied radar systems have had varying image quality. As a reference, the processor was found to work very well with simulated data. The results obtained with the UAVSAR data were also of high quality, with no significant motion related errors. The UMass S-band system results were of medium quality, with varying levels of motion related errors, such that some scenes focused well and others poorly. In contrast, the Ka-band system had generally poor results, with almost all scenes having significant motion related focusing errors. As a first-

order correction to the motion errors for the UMass systems, a simple offset to the measured yaw and pitch angles was calculated. This was done using the above-defined procedure, fitting to the squint model from the Doppler spectrum as well as using a simple grid search of offsets and comparing the focusing quality subjectively. Due to the high sensitivity of the Ka-band system to attitude errors, both methods worked equally well. It should be noted that no correction to position was estimated. Using this first-order correction, both the S-band and Ka-band imagery showed generally good results, though both still contained motion-related errors. These errors manifested as minor defocusing when the platform was moving fairly linearly, and more significant defocusing, as the linearity of the flight path degraded.

These results are not unexpected due to the fact that the UAVSAR system generally flies quite straight and has a high quality motion measurement system. The UMass radars fly on a small, human piloted, low-altitude aircraft and currently use a much less sophisticated motion measurement system.

For the UAVSAR system, further investigation is needed to determine what the dominant source of error is before corrections can be added. For the UMass systems there are clear improvements that could be made in both the position and attitude measurement systems (again it is noted that these are truly coupled). While it may be possible to correct for the motion measurement errors using some combination of the above estimation methods, the motion measurement system is in the process of being updated, and further investigation will continue after the upgrade is complete. This upgrade includes a more accurate INU and GPS receiver, capable of performing coupled post-processing using the raw INU and GPS measurements as well as *a posteriori* GPS ephemeris and atmospheric models. In addition, the upgrade will include a more accurate survey measurement of the relative positions of the INU, GPS and radar antennas on the platform. The upgrade should improve the results

significantly on its own, and should reduce the burden on any motion estimation procedure.

5.4 Summary

This chapter provides an overview of the implementation of a backprojection processor. The nature of time-domain backprojection makes it well suited for implementation with graphics processing units, which are designed to be efficient when performing highly parallel computations. A description of GPU hardware is given, with specifics relevant to this implementation. Some steps important to backprojection processing are first presented here, including the NERFFT interpolation step as applied to the pulse-compressed data. Motion estimation is presented as an auxiliary section, as it may be another important processing step for systems with limited motion measurement accuracy. The following chapter discusses the various coordinate systems in use by the processor, and how conversions are carried out between them.

CHAPTER 6

COORDINATE SYSTEMS

There are several different coordinate systems in use by the processor. This chapter serves as a reference to them and their relations.

The core of the processing uses the earth-centered, earth-fixed (ECEF) Cartesian[1] coordinate system. Here, the ECEF coordinate system is also referred to as the XYZ system. This system is a right-handed Cartesian coordinate system centered at the center of the earth's mass. The x-axis points toward the intersection of the prime meridian and the equator. The y-axis points toward the intersection of the 90 degrees longitude and the equator. Thus the z-axis points (approximately) toward the north pole.

Commonly, an ellipsoid is used to approximate the shape of the earth and is also centered at the earth's center of mass [21]. Here, the WGS84 ellipsoid is used. This ellipsoid is what is commonly used to define latitude and longitude angles. In reality, the WGS84 ellipsoid is actually a spheroid (or ellipsoid of revolution), which has two distinct radii known as the semi-major axis and semi-minor axis (a true ellipsoid has three distinct radii). To follow convention this spheroid will be referred to as an ellipsoid in further discussion. The ellipsoid can be described as [21]

$$\frac{x^2 + y^2}{a} + \frac{z^2}{b} = 1 \tag{6.1}$$

[1] A three dimensional Cartesian system defines points using three fixed, orthogonal axes. This is in contrast to non-Cartesian systems like spherical or polar coordinates.

where a corresponds to the semi-major axis and b to the semi-minor axis. The ellipsoid may also be described by the semi-major axis and the eccentricity squared, e^2, where

$$e^2 = 1 - \frac{b^2}{a^2} \tag{6.2}$$

When describing the Earth, the semi-major axis is the equatorial radius and the semi-minor axis is the polar radius.

A point on the surface of the ellipsoid is normally defined by latitude and longitude angles. As the ellipsoid is just an ellipse rotated about the z-axis, the longitude angle, θ, for a given point is simply this rotation about the z-axis. In the vertical plane the latitude angle is described in one of two ways. Geocentric latitude, ϕ, is the angle between the radius from the earth's center to the point and the equatorial plane. Geodetic latitude, λ, is the angle between the surface normal at the point and the equatorial plane. Since the shape is an ellipse in the vertical axis, the surface normal does not correspond with the radius for any given point, so the geocentric latitude will not be the same as the geodetic latitude. The geodetic latitude is what is usually referred to and used. Additionally, the height above the surface, h, is defined along the surface normal.

There are several distinct radii that may be used with a given surface point [21]. There is the radius extending from the earth's center to the surface point, r. To describe the amount of curvature of the surface at the given point, two other radii are used. The east-west radius of curvature, r_e, describes the radius of a sphere whose curvature matches the curvature of the surface at the given point, in the east-west direction. It is defined as

$$r_e(\lambda) = \frac{a}{\sqrt{1 - e^2 \sin^2(\lambda)}}. \tag{6.3}$$

In the north-south direction a similar radius of curvature exists, r_n, defined as

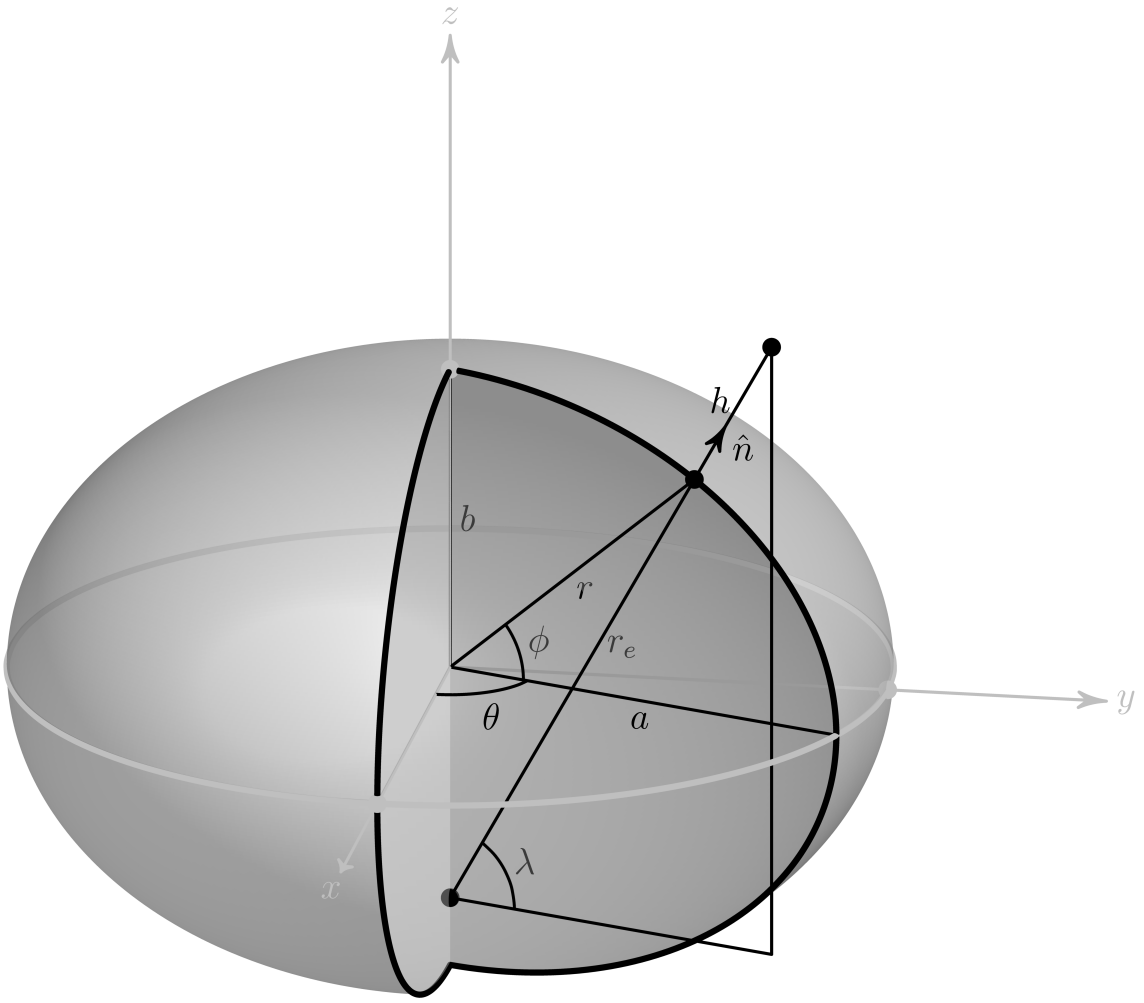


Figure 6.1: Model of the earth ellipsoid. Note that for clarity the eccentricity of the ellipsoid shown here is far exaggerated relative to the WGS84 ellipsoid.

$$r_n(\lambda) = \frac{a(1-e^2)}{\left(1-e^2\sin^2(\lambda)\right)^{\frac{3}{2}}}. \tag{6.4}$$

Two Cartesian systems are commonly used to approximate the surface of the earth at a given point. Both the East, North, Up (ENU) and North, East, Down (NED) coordinate systems provide useful right-handed Cartesian reference systems for any point on the globe. In each, the first two axes are defined to match the local north or east vectors on the surface. The third axis is either defined to match (positively or negatively) the surface normal or the earth center radius. Here, the surface normal is used. Note that either of these coordinate systems may commonly be located at any altitude, and are usually defined to be co-located with the center of a vehicle of interest. In this case, the approximation is to the surface directly below (in either convention) the platform. Additionally, either coordinate system can be rotated about the up/down axis to match the heading of a platform. The rotated NED system is referred to as the Track, Cross-track, Nadir (TCN) system. Further, this system may be related to the full orientation of the vehicle, represented in the NED/TCN case by the "*ijk*" system, representing the forward, right, and down axes in the vehicle frame. Following the ENU convention the respective coordinate systems would be the Track, Cross-track, Up (TCU) system and the "IJK" forward, left, up system.

To best account for the curvature of the earth's surface over large radar swaths, while maintaining relative simplicity, the SCH coordinate system was developed [13]. The SCH coordinate system is a spherical coordinate system designed to match the curvature of the ellipsoid at a given reference point known as the peg point, and in a given direction. This spherical approximation can be seen as a compromise between using a flat Cartesian approximation at a given point on the surface, which is simple to use but not very accurate, and the full ellipsoidal approximation, which is quite accurate but difficult to use. The SCH system is based on the ability to describe the

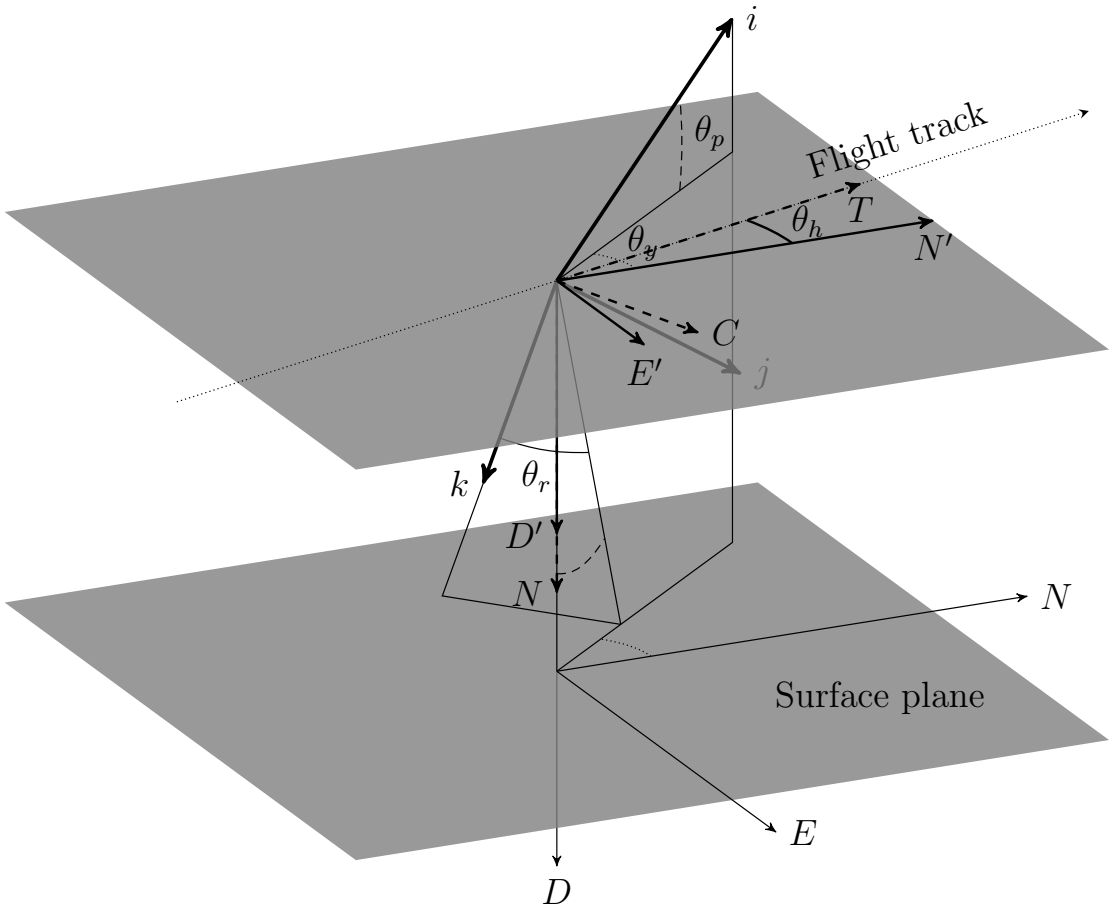


Figure 6.2: Platform coordinate systems.

radius of curvature of the ellipsoid in an arbitrary direction at a given point by

$$r_a = \frac{r_e(\lambda) r_n(\lambda)}{r_e(\lambda)\cos^2(\eta) + r_n(\lambda)\sin^2(\eta)} \tag{6.5}$$

where the geodetic heading is given by η. The SCH coordinate system uses a sphere of radius r_a, and center defined such that the surface tangent plane at the peg point, on the ellipsoid, corresponds with the surface tangent plane of a point on the sphere. Defined in this way, the sphere will be the closest spherical approximation to the curvature of the ellipsoid along some given track. This is ideal for use by stripmap radars, as in this system the error increases most slowly along the given heading, where the majority of the extent of the area of interest will be. The sphere-centered Cartesian coordinates, (x', y', z'), are defined such that the x'-axis corresponds to the surface normal at the peg point, and the heading vector at the peg point lies within the equatorial plane. Thus, the heading vector will always be aligned with the sphere's equator, in the direction of decreasing longitude.[2] A right-handed coordinate system with axes $\hat{s}\hat{c}\hat{h}$ can be defined at any point on the sphere and represents the local tangent plane. At the peg point, $\hat{s}$ is coincident with the heading vector, $\hat{h}$ along the surface normal and $\hat{c}$ defined to complete the system. In the SCH coordinate systems points are described using their distance along the equator, s, the distance along a meridian perpendicular to this point, c, and finally a height, h, normal to the surface at the point (s, c). It should be noted that the coordinates (s, c, h) are not Cartesian, though over small distances are approximately the same as the $\hat{s}\hat{c}\hat{h}$ Cartesian system.

A common coordinate system used with radar systems is a two-dimensional system known as slant-range coordinates or radar coordinates. The two dimensions in this system are azimuth and range, representing distance horizontally along a refer-

[2]Longitude referring to the longitude on the sphere, not the ellipsoid longitude.

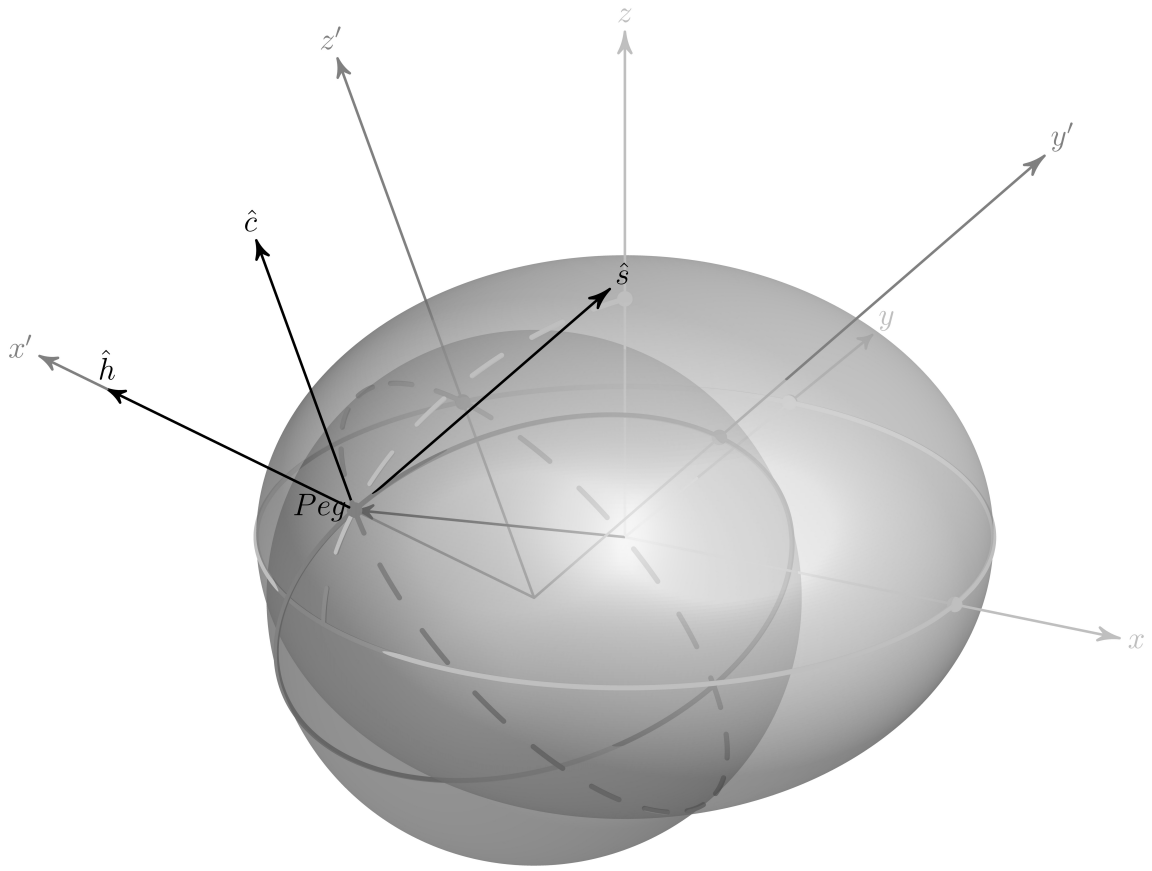


Figure 6.3: A representation of the SCH approximating sphere. Note that for clarity the eccentricity of the ellipsoid shown here is far exaggerated relative to the WGS84 ellipsoid.

ence flight track and distance radially away from the reference track. A point with coordinates (a, r) is a units displaced from the reference track origin along the track and r units radially displaced from the point $(a, 0)$. Generally, in three-dimensions, these coordinates would be ambiguous, as a point at (a, r) could be located anywhere on a circle with origin $(a, 0)$ and radius r. In practice, such a coordinate system is often used above a smooth surface, and (a, r) points are assumed to lie on that surface. Here the (a, r) specification could represent zero, one or two intersections with the surface. Ignoring points above the surface (zero intersections), and assuming a radar system to look in only one direction (left or right) from the reference track, constrains the system to an unambiguous single intersection between an (a, r) point and the surface.

A reference track for radar coordinates may be specified in a Cartesian system, such as the TCN system, or alternatively, in a non-Cartesian system such as SCH. The SCH system is used here. In the SCH system, a reference track is specified by its peg point, including the latitude, longitude and heading, as well as an altitude. Thus, the (a, r) coordinates represent distance along the spherical reference track and radial distance away from the track and will be alternately specified by (s, r) to match the SCH system. The spherical surface of the SCH system enables non-ambiguous location specification for radar systems. Since the track and surface are not linear, the (s, r) coordinates will not be Cartesian.

6.1 Rotations and Conversions

Several types of coordinate conversions are relevant to this processor and may be separated into three categories. The first category involves conversions between Cartesian systems that share the same origin but whose axes point in different directions. An example of this rotation would be a conversion between platform body *ijk* coordinates and the platform centered north-east-down, NED, coordinates. The

second category involves conversions between Cartesian systems that do not share the same origin, and whose axes are most often not parallel. Conversion from the NED coordinates and ECEF XYZ coordinates is an example. The third category is more broad and encompasses all conversions involving at least one non-Cartesian system. Conversion from latitude-longitude-height to ECEF XYZ coordinates falls into this category.

The relative orientation of two Cartesian coordinate systems is usually defined using a rotation. In this sense, the coordinates of a point are said to be rotated between the coordinate systems. Often, this convention is used to describe the orientation of a moving platform, which may have literally rotated into its current position relative to some reference system. These rotations may be described using Euler (or similar) angles, rotation matrices, or quaternions, all three of which are used in this processor.

Euler angles describe the relative orientation of two systems using three angles, referred to here as yaw (θ), pitch (ϕ), and roll (γ) [2]. These angles represent three successive rotations that may be applied to the axes of a reference system, which result in the orientation of the second system. Each successive rotation is applied about one axis in a direction corresponding to clockwise when looking out along the axis from the origin. The order of these rotations and the axes they are applied to are a matter of convention. Two common conventions use the order z-y-x, or z-y-z. Another convention defines whether the axes used for the rotations are the axes of the reference coordinate system, or the axes of the intermediate coordinate system, which has resulted from the previous rotations. Here, the second convention is used. For example, using the order z-y-x and intermediate axes a rotation would be carried out as follows. The reference coordinate system (x, y, z) is rotated about its z-axis clockwise by the yaw angle to arrive with the intermediate system (x', y', z'). The (x', y', z') system is rotated clockwise about its y'-axis by the pitch angle to arrive with the intermediate system (x'', y'', z''). Finally the (x'', y'', z'') system is rotated

clockwise about its x''-axis by the roll angle to arrive at the final (X, Y, Z) system. The coordinates of a single, fixed point, may be transformed between the two systems by applying such a rotation scheme.

In practice, the transformation, or rotation, of a point's coordinates between two systems is carried out using rotation matrices. In three dimensions, a rotation can applied using a 3×3 matrix. For example, when rotating from reference coordinates (x, y, z) to final coordinates (x', y', z')

$$\bar{p}_{x'y'z'} = \begin{bmatrix} p_{x'} \\ p_{y'} \\ p_{z'} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} = \mathbf{I}\bar{p}_{xyz} \tag{6.6}$$

represents no rotation. Rotation by angle θ about the x-axis is given by

$$\bar{p}_{x'y'z'} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & \sin(\theta) \\ 0 & -\sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} = \mathbf{X}(\theta)\bar{p}_{xyz}. \tag{6.7}$$

Similarly y-and z-axis rotations are

$$\bar{p}_{x'y'z'} = \begin{bmatrix} \cos(\theta) & 0 & -\sin(\theta) \\ 0 & 1 & 0 \\ \sin(\theta) & 0 & \cos(\theta) \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} = \mathbf{Y}(\theta)\bar{p}_{xyz} \tag{6.8}$$

$$\bar{p}_{x'y'z'} = \begin{bmatrix} \cos(\theta) & \sin(\theta) & 0 \\ -\sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} = \mathbf{Z}(\theta)\bar{p}_{xyz} \tag{6.9}$$

respectively. From these definitions, it is possible to build a single rotation matrix representing the Euler rotations. The intrinsic z-y-x rotation has the following form

$$\begin{aligned}\bar{p}_{XYZ} =& \begin{bmatrix} \mathrm{c}(\theta)\,\mathrm{c}(\phi) & \mathrm{c}(\theta)\,\mathrm{s}(\phi)\,\mathrm{s}(\gamma) + \mathrm{s}(\theta)\,\mathrm{c}(\gamma) & -\mathrm{c}(\theta)\,\mathrm{s}(\phi)\,\mathrm{c}(\gamma) + \mathrm{s}(\theta)\,\mathrm{s}(\gamma) \\ -\mathrm{s}(\theta)\,\mathrm{c}(\phi) & -\mathrm{s}(\theta)\,\mathrm{s}(\phi)\,\mathrm{s}(\gamma) + \mathrm{c}(\theta)\,\mathrm{c}(\gamma) & \mathrm{s}(\theta)\,\mathrm{s}(\phi)\,\mathrm{c}(\gamma) + \mathrm{c}(\theta)\,\mathrm{s}(\gamma) \\ \mathrm{s}(\phi) & -\mathrm{c}(\phi)\,\mathrm{s}(\gamma) & \mathrm{c}(\phi)\,\mathrm{c}(\gamma) \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} \\ =& \mathbf{Z}(\theta)\mathbf{Y}(\phi)\mathbf{X}(\gamma)\bar{p}_{xyz} = \mathbf{R}_{xyz}^{XYZ}(\theta,\phi,\gamma)\bar{p}_{xyz} \end{aligned} \tag{6.10}$$

where $\mathrm{c}(a) \triangleq \cos(a)$ and $\mathrm{s}(a) \triangleq \sin(a)$ are defined for convenience.

A convention is established here for representing rotations between coordinate systems with rotation matrices. A rotation matrix which will rotate coordinates from frame abc to frame $a'b'c'$ is represented by $\mathbf{R}_{abc}^{a'b'c'}$. The coordinates of a point in the frame $a'b'c'$ are given as $\bar{p}_{a'b'c'}$ and can be generated by right-multiplying the rotation matrix by the coordinates in frame abc, written as

$$\bar{p}_{a'b'c'} = \mathbf{R}_{abc}^{a'b'c'}\bar{p}_{abc}. \tag{6.11}$$

The inverse rotation, generating the coordinates of a point in frame abc from the $a'b'c'$ coordinates would be written

$$\bar{p}_{abc} = \mathbf{R}_{a'b'c'}^{abc}\bar{p}_{a'b'c'}. \tag{6.12}$$

The rotation matrix representing the inverse rotation is also the inverse of the forward rotation matrix. Furthermore, since rotation matrices are orthogonal the two matrices are also related by their transpose, thus

$$\mathbf{R}_{a'b'c'}^{abc} = \mathbf{R}_{abc}^{a'b'c'^{-1}} = \mathbf{R}_{abc}^{a'b'c'^{T}}. \tag{6.13}$$

It should be noted that such rotation matrices may be multiplied together to perform multiple conversions, as

$$\bar{p}_{abc} = \mathbf{R}^{abc}_{mno}(\mathbf{R}^{mno}_{a'b'c'}\bar{p}_{a'b'c'}) = (\mathbf{R}^{abc}_{mno}\mathbf{R}^{mno}_{a'b'c'})\bar{p}_{a'b'c'} = \mathbf{R}^{abc}_{a'b'c'}\bar{p}_{a'b'c'} \tag{6.14}$$

While Euler angles are intuitive and often convenient to use, they suffer from ambiguities for some angles. In the case above when $\phi = \pm\frac{\pi}{2}$, the two angles θ and γ represent the same rotation. This ambiguity in the specification of θ and γ is know as gimbal lock. To avoid this problem, the full nine element matrix may be used, which is unambiguous. Alternatively, another unambiguous method of defining rotations may be used, requiring four elements [11]. Euler's rotation theorem states that any coordinate system rotation of the type under discussion may be represented by rotation about a unit vector $\bar{u}$ by a single angle θ. Here the four elements are the three coordinates of the vector and the angle θ. Typically, quaternions are used to represent these rotations. A quaternion in general is an extension to the complex number set in which "quaternions" are represented by $a + b\mathbf{i} + c\mathbf{j} + d\mathbf{k}$ where a, b, c and d are real, scalar numbers and $\mathbf{i}$, $\mathbf{j}$ and $\mathbf{k}$ are quaternion units (analogous to the imaginary unit $\mathbf{j}$ for standard complex numbers) [12]. Further, the units $\mathbf{i}$, $\mathbf{j}$ and $\mathbf{k}$ satisfy $\mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 = \mathbf{ijk} = -1$. A rotation by angle θ about vector $\bar{u}_{xyz}$ can be represented by the quaternion

$$\bar{q} = \cos\left(\frac{\theta}{2}\right) + (u_x\mathbf{i} + u_y\mathbf{j} + u_z\mathbf{k})\sin\left(\frac{\theta}{2}\right). \tag{6.15}$$

With this definition, the rotation can be applied as

$$\bar{p}^q_{x'y'z'} = \bar{q}\bar{p}^q_{xyz}\bar{q}^{-1} \tag{6.16}$$

where $\bar{p}^q_{xyz}$ is a quaternion such that

$$\bar{p}^q_{xyz} = 0 + (p_x\mathbf{i} + p_y\mathbf{j} + p_z\mathbf{k}). \tag{6.17}$$

A quaternion of the form $a+b\mathbf{i}+c\mathbf{j}+d\mathbf{k}$ may be converted to a rotation matrix as [11]

$$\mathbf{Q} = \begin{bmatrix} a^2+b^2-c^2-d^2 & 2(bc+ad) & 2(bd-ac) \\ 2(bc-ad) & a^2-b^2+c^2-d^2 & 2(cd+ab) \\ 2(bd+ac) & 2(cd-ab) & a^2-b^2-c^2+d^2 \end{bmatrix}. \tag{6.18}$$

The opposite conversion, from a rotation matrix to a quaternion, is not as straightforward, but a robust algorithm is described by Bar-Itzhack [1]. The method described constructs the 4×4 matrix

$$\mathbf{K} = \frac{1}{3}\begin{bmatrix} \mathbf{R}_{00}-\mathbf{R}_{11}-\mathbf{R}_{22} & \mathbf{R}_{10}+\mathbf{R}_{01} & \mathbf{R}_{02}+\mathbf{R}_{02} & \mathbf{R}_{12}-\mathbf{R}_{01} \\ \mathbf{R}_{10}+\mathbf{R}_{01} & \mathbf{R}_{11}-\mathbf{R}_{00}-\mathbf{R}_{22} & \mathbf{R}_{01}+\mathbf{R}_{12} & \mathbf{R}_{02}-\mathbf{R}_{02} \\ \mathbf{R}_{02}+\mathbf{R}_{02} & \mathbf{R}_{01}+\mathbf{R}_{12} & \mathbf{R}_{22}-\mathbf{R}_{00}-\mathbf{R}_{11} & \mathbf{R}_{01}-\mathbf{R}_{10} \\ \mathbf{R}_{12}-\mathbf{R}_{01} & \mathbf{R}_{02}-\mathbf{R}_{02} & \mathbf{R}_{01}-\mathbf{R}_{10} & \mathbf{R}_{00}+\mathbf{R}_{11}+\mathbf{R}_{22} \end{bmatrix} \tag{6.19}$$

from the rotation matrix $\mathbf{R}$. Finding the eigenvector corresponding to the largest eigenvalue of this matrix yields the quaternion. It is shown that even when the rotation matrix contains accumulated numerical error (and is therefore not orthogonal and not truly a rotation matrix), the quaternion found using this method is valid and represents the rotation which is closest to the non-orthogonal matrix given.

It should be noted that there are exactly two quaternions which may be used to represent one rotation: one derived from $\bar{u}$ and θ, as well as one derived from $-\bar{u}$ and $-\theta$. Due to this fact, and the errors which may exist in any given rotation matrix, quaternions resulting from the Bar-Itzhak algorithm may be nearly opposite from each other, despite representing similar rotations. When representing a time-series of coordinate rotations, successive quaternions may be negated if they are

nearly opposite from previous ones. This ensures that the numerical values of the quaternions will change slowly, as long as the actual rotation is also changing slowly.

To summarize, a rotational offset between coordinate systems may be represented by either a set of three Euler angles, a nine-element rotation matrix or a four-element quaternion. To apply the rotation (i.e. rotate the coordinates of a given point from one coordinate system to another) the rotation matrix or quaternion may be used. It is possible to translate between each representation with minor caveats.

In addition to pure rotations coordinate systems may also be displaced. Representing both translation and rotation of two Cartesian coordinate systems can be achieved using one of the above rotation representations and a simple translation vector. Thus a full rotation and translation conversion from frame abc to frame $a'b'c'$ can be carried out as

$$\bar{p}_{a'b'c'} = \mathbf{R}_{abc}^{a'b'c'} \bar{p}_{abc} + \bar{T}_{abc}^{a'b'c'} \tag{6.20}$$

where the translation vector is defined in the coordinates of the final system ($a'b'c'$ in this case). The vector $\bar{T}_{abc}^{a'b'c'}$ points from the $a'b'c'$ origin to the abc origin, in the $a'b'c'$ system (it is the position of the abc origin in $a'b'c'$ coordinates). Given this convention the same conversion can be represented as

$$\bar{p}_{a'b'c'} = \mathbf{R}_{abc}^{a'b'c'} \left(\bar{p}_{abc} - \bar{T}_{a'b'c'}^{abc} \right) \tag{6.21}$$

or the inverse conversion as

$$\bar{p}_{abc} = \mathbf{R}_{a'b'c'}^{abc} \bar{p}_{a'b'c'} + \bar{T}_{a'b'c'}^{abc} \tag{6.22}$$

or

$$\bar{p}_{abc} = \mathbf{R}_{a'b'c'}^{abc} \left(\bar{p}_{a'b'c'} - \bar{T}_{abc}^{a'b'c'} \right). \tag{6.23}$$

Given matrix and vector properties, the two translation vectors can be related by

$$\bar{T}^{abc}_{a'b'c'} = -\mathbf{R}^{abc}_{a'b'c'}\bar{T}^{a'b'c'}_{abc} \tag{6.24}$$

or

$$\bar{T}^{a'b'c'}_{abc} = -\mathbf{R}^{a'b'c'}_{abc}\bar{T}^{abc}_{a'b'c'}. \tag{6.25}$$

For convenience, the following notation will be used to represent a general coordinate conversion

$$\bar{p}_{a'b'c'} = \mathcal{T}^{a'b'c'}_{abc}\{\bar{p}_{abc}\} \tag{6.26}$$

where the conversion operator $\mathcal{T}$ may involve many rotations, translations and/or any of the non-standard conversions listed below.

For non-Cartesian coordinate systems, unique conversions must be used. Those relevant are described below. Given the definitions of the earth ellipsoid a conversion between latitude, longitude and height (l, l, h) coordinates and ECEF (x, y, z) coordinates is [21]

$$\bar{p}_{XYZ} = \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} = \begin{bmatrix} (r_e(\lambda) + h)\cos(\lambda)\cos(\theta) \\ (r_e(\lambda) + h)\cos(\lambda)\sin(\theta) \\ (r_e(\lambda)(1 - e^2) + h)\sin(\lambda) \end{bmatrix} \tag{6.27}$$

An exact conversion from ECEF coordinates to LLH coordinates is given by Vermeille [23]. The algorithm computes in order the following:

$$
\begin{aligned}
p &= \frac{X^2 + Y^2}{a^2} \\
q &= \frac{1 - e^2}{a^2} Z^2 \\
r &= \frac{p + q - e^4}{6} \\
s &= e^4 \frac{pq}{4r^3} \\
t &= \sqrt[3]{1 + s + \sqrt{s(2+s)}} \\
u &= r\left(1 + t + \frac{1}{t}\right) \\
v &= \sqrt{u^2 + e^4 q} \\
w &= e^2 \frac{u + v - q}{2v} \\
k &= \sqrt{u + v + w^2} - w \\
D &= \frac{k\sqrt{X^2 + Y^2}}{k + e^2} \\
\theta &= 2 \arctan\left(\frac{Y}{X + \sqrt{X^2 + Y^2}}\right) \\
\lambda &= 2 \arctan\left(\frac{Z}{D + \sqrt{D^2 + Z^2}}\right) \\
h &= \frac{k + e^2 - 1}{k} \sqrt{D^2 + Z^2}.
\end{aligned} \tag{6.28}
$$

While a bit complex, this algorithm gives an exact solution which is stable at the poles. Though just a standard rotation and translation, the conversion between the surface tangent representation ENU and ECEF can be described using

$$
\mathbf{R}_{enu}^{xyz} = \begin{bmatrix} -\sin(\theta) & -\sin(\lambda)\cos(\theta) & \cos(\lambda)\cos(\theta) \\ \cos(\theta) & -\sin(\lambda)\sin(\theta) & \cos(\lambda)\sin(\theta) \\ 0 & \cos(\lambda) & \sin(\lambda) \end{bmatrix} \tag{6.29}
$$

and the translation vector, generated using equation (6.27), giving the conversion

$$
\bar{p}_{xyz} = \mathbf{R}_{enu}^{xyz} \bar{p}_{enu} + \bar{T}_{enu}^{xyz}. \tag{6.30}
$$

There are several simple rotations matrices which describe conversions between similar coordinate systems. Rotating between *ijk* and IJK coordinates, or TCN and TCU coordinates uses the rotation matrix

$$\mathbf{R}_{ijk}^{IJK} = \mathbf{R}_{IJK}^{ijk} = \mathbf{R}_{TCN}^{TCU} = \mathbf{R}_{TCU}^{TCN} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix}. \tag{6.31}$$

Rotating between ENU and NED coordinates uses the rotation matrix

$$\mathbf{R}_{ENU}^{NED} = \mathbf{R}_{NED}^{ENU} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & -1 \end{bmatrix}. \tag{6.32}$$

Finally, rotating between a right-handed to left-handed coordinate system (i.e forward, right, down to forward, left, down) uses

$$\mathbf{R}_{\text{Right}}^{\text{Left}} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix}. \tag{6.33}$$

A number of conversions are in use relating to the SCH coordinate system, as described in [13]. The center of the SCH refrence sphere is calculated using the (x, y, z) coordinates of the peg point, and the surface normal vector, $\bar{h}$, as

$$\bar{c}_{xyz} = \bar{p}_{\text{peg},xyz} - r_a \bar{h} \tag{6.34}$$

or

$$\begin{bmatrix} c_x \\ c_y \\ c_z \end{bmatrix} = \begin{bmatrix} r_e \cos(\lambda)\cos(\theta) \\ r_e \cos(\lambda)\sin(\theta) \\ r_e(1-e^2)\sin(\lambda) \end{bmatrix} - r_a \begin{bmatrix} \cos(\lambda)\cos(\theta) \\ \cos(\lambda)\sin(\theta) \\ \sin(\lambda) \end{bmatrix} = \begin{bmatrix} (r_e - r_a)\cos(\lambda)\cos(\theta) \\ (r_e - r_a)\cos(\lambda)\sin(\theta) \\ (r_e(1-e^2) - r_a)\sin(\lambda) \end{bmatrix}. \tag{6.35}$$

Conversion from SCH based (x', y', z') coordinates to ECEF coordinates uses the rotation matrix

$$\mathbf{R}^{ENU}_{x'y'z'} = \begin{bmatrix} 0 & \sin(\eta) & -\cos(\eta) \\ 0 & \cos(\eta) & \sin(\eta) \\ 1 & 0 & 0 \end{bmatrix} \tag{6.36}$$

The conversion is then carried out by

$$\bar{p}_{xyz} = \mathbf{R}^{xyz}_{ENU}\mathbf{R}^{ENU}_{x'y'z'}\bar{p}_{x'y'z'} + \bar{T}^{xyz}_{x'y'z'} = \mathbf{R}^{xyz}_{x'y'z'}\bar{p}_{x'y'z'} + \bar{T}^{xyz}_{x'y'z'} \tag{6.37}$$

where $\bar{T}^{xyz}_{x'y'z'}$ is generated from (6.34). Converting between the quasi-spherical (s, c, h) coordinates and (x', y', z') coordinates uses the relation

$$\bar{p}_{x'y'z'} = (r_a + p_h) \begin{bmatrix} \cos\left(\frac{p_c}{r_a}\right)\cos\left(\frac{p_s}{r_a}\right) \\ \cos\left(\frac{p_c}{r_a}\right)\sin\left(\frac{p_s}{r_a}\right) \\ \sin\left(\frac{p_c}{r_a}\right) \end{bmatrix} \tag{6.38}$$

or

$$\bar{p}_{sch} = \begin{bmatrix} r_a \arctan\left(\frac{p_{y'}}{p_{x'}}\right) \\ r_a \arcsin\left(\frac{p_{z'}}{\sqrt{p^2_{x'}+p^2_{y'}+p^2_{z'}}}\right) \\ \sqrt{p^2_{x'} + p^2_{y'} + p^2_{z'}} - r_a \end{bmatrix}. \tag{6.39}$$

Rotation from the local sphere-tangent $(\hat{s}, \hat{c}, \hat{h})$ coordinates, defined at some point $\bar{p}_{sch}$, to the (x', y', z') coordinates is described by

$$\mathbf{R}^{x'y'z'}_{\hat{s}\hat{c}\hat{h}} = \begin{bmatrix} -\sin\left(\frac{p_s}{r_a}\right) & -\sin\left(\frac{p_c}{r_a}\right)\cos\left(\frac{p_s}{r_a}\right) & \cos\left(\frac{p_c}{r_a}\right)\cos\left(\frac{p_s}{r_a}\right) \\ \cos\left(\frac{p_s}{r_a}\right) & -\sin\left(\frac{p_c}{r_a}\right)\sin\left(\frac{p_s}{r_a}\right) & \cos\left(\frac{p_c}{r_a}\right)\sin\left(\frac{p_s}{r_a}\right) \\ 0 & \cos\left(\frac{p_c}{r_a}\right) & \sin\left(\frac{p_c}{r_a}\right) \end{bmatrix}. \tag{6.40}$$

The two-dimensional radar coordinates can be converted to other systems given unambiguous specification. When specified in the SCH system the (s, r) radar coordinates can be converted using the law of cosines as

$$\bar{p}_{sch} = \begin{bmatrix} s \\ (\mathrm{LR}) r_a \arccos\left(\frac{r^2-(r_a+\alpha)^2-(r_a+h)^2}{-2(r_a+\alpha)(r_a+h)}\right) \\ h \end{bmatrix} \tag{6.41}$$

where $(\mathrm{LR}) = 1$ for left looking radars and -1 for right looking radars, α is the height of the reference track, and h is the height of the point being converted. By specifying some $h \neq 0$ the coordinates of a point not on the spherical surface can be converted, allowing the use of a non-smooth surface such as a DEM. The inverse conversion can be carried out similarly, as in

$$\bar{p}_{sr} = \begin{bmatrix} s \\ \sqrt{(r_a+\alpha)^2 + (r_a+h)^2 - 2(r_a+\alpha)(r_a+h)\cos\left(\frac{c}{r_a}\right)} \end{bmatrix}. \tag{6.42}$$

Using the definitions above, coordinate conversions can be carried out between any of the defined systems by appropriately combining different conversions together. One example is converting the coordinates of a point specified in (x, y, z) coordinates to the local (i, j, k) coordinates given of a vehicle whose location is specified in (x, y, z) coordinates. The conversion would have the following form

$$\bar{p}_{ijk} = \mathbf{R}_{ijk}^{NED^T} \mathbf{R}_{ENU}^{NED} \mathbf{R}_{xyz}^{ENU} \bar{p}_{xyz} + \bar{T}_{x'y'z'}^{xyz} = \mathbf{R}_{x'y'z'}^{ijk} \bar{p}_{x'y'z'} + \bar{T}_{x'y'z'}^{xyz} \tag{6.43}$$

6.2 Summary

This chapter provides a detailed look into the coordinate systems relevant to the processor, of which there are several types of Cartesian and non-Cartesian coordinate

systems. A few representations were given for global scale coordinates, including a local spherical surface approximation known as SCH, which is well suited for use with stripmap radar systems. There were also many local scale coordinate systems mentioned, notably Cartesian surface representations, platform coordinates and radar coordinates. The last section was devoted to mathematical descriptions of how to convert between the relevant coordinate systems, including three ways of representing rotations, as well as conversions for non-Cartesian coordinates. Following from the implementation details provided in this and the previous chapter, a step by step look into the entire processing chain is given next.

CHAPTER 7

PROCESSOR DESCRIPTION

What is presented here is intended to provide the clearest understanding of how this processor works. In this discussion, the processor chain is defined to start with the data inputs, which have been recorded and made ready for the processor, and end with the focused SAR data outputs. Some system specific processing is done prior to running the processor, which includes any data format handling and potential GPS/INU motion post-processing. In the case of UAVSAR, the raw data is pulse-compressed externally from this processor, using existing tools. It should be noted that a few of the processing steps mentioned below (specifically the target grid height calculation and radar state file handling) use external tools developed by Brian Hawkins of JPL. The block diagram shown in Figure 5.2 is a useful reference here.

7.1 Preprocessor

There are a few basic tasks performed with the raw radar system data, beyond what is mentioned above, prior to the processing itself. The goal of this preprocessor is to generate a "Radar State" file which contains all position, attitude and additional metadata information for each transmitted radar pulse. Additionally, the preprocessor will generate some configuration information used by the processor, most notably the best fit SCH reference track.

The first step of the preprocessing is to read the timestamps for each radar pulse within the given file. For convenience, all times are converted to a local time scale using the transmission time of the first pulse as zero. Next, the GPS positions and

INU attitude information is read, including enough entries to cover the entire scene. The timestamps for each are converted to the local scale. The INU attitude angles are converted to quaternion rotations describing the angular offset between the antenna and the ECEF XYZ coordinate system. A cubic interpolator is generated to allow sampling of the attitude data at all times within the scene. The parameters of this interpolator can be adjusted to enable smoothing of the data. This interploator will become part of the radar state file, and will be used to calculate the antenna attitude for every pulse. It is represented in the file by its interpolation coefficients.

Before doing the same with the GPS positions, one further task must be completed. The positions given by the GPS receiver are usually the positions of the antenna phase center of the GPS antenna, or alternatively the INU reference position if these data are blended. In either case, these positions must be translated to the positions of the antenna phase center of the *radar* antenna, which are the desired positions for the processor. This is done by assuming the platform body is rigid and using a fixed lever arm (in the platform's coordinate system) describing the offset between the radar antenna and GPS reference point. Since the attitude of the platform varies the absolute offset, in any global coordinate system, will vary (even though the offset is fixed in the platform's coordinate system).

In order to properly translate the GPS positions, which are represented as ECEF XYZ positions, the lever arm is rotated into the ECEF system using the platform attitude. The rotated lever arm is applied to the GPS positions to translate them to radar antenna positions. From this point, a similar cubic interpolator is generated for the radar antenna positions, in the ECEF XYZ system.

The radar state is written to disk. It contains both position and attitude interpolators as well as the timestamps for each pulse, and the per-pulse metadata. Some parameters are calculated and written to an output configuration file to aid in the generation of the configuration for the processor itself. Most notably, these include

a best fit track to the radar antenna positions, denoted as an SCH peg point, and track altitude. The position of the first pulse, in the fit SCH coordinate system, is also written.

The radar state generated by the preprocessor contains all of the information necessary to run the processing for any desired scene which uses the radar pulses within it. This enables the processor to be run multiple times, with different configurations, without having to perform the tasks of the preprocessor each time.

7.2 Processor

The first step of the processor is to read the configuration file. This includes relevant radar parameters, general processor options, output grid specification and input/output file names.

The radar state file is read, giving position, attitude, timing and extra parameters for each pulse. These parameters include a system gain, electronic steering angle, delay until reception and a data type flag. The position, $\bar{p}_{n,t,xyz}$, is stored as three separate cubic splines for each of the x, y, and z ECEF coordinates of the antenna position. The antenna attitude is stored as four separate cubic splines for each of the four quaternion units representing a rotation from the antenna-centered system to ECEF coordinates. When evaluated, the position and attitude are sampled at the transmission time given for each pulse, resulting in $\bar{p}_{n,0,xyz}$ and the attitude similarly sampled.

The antenna-centered system is a right handed system with axes pointing nominally toward forward, down and left (FDL). When accessed the quaternion is converted into a rotation matrix, represented by $\mathbf{R}_{fdl}^{xyz}$. A single entry in the state file stores the physical elevation angle between the antenna-centered system and the actual antenna boresight, θ_b. For a left-looking antenna, rotating the FDL system about the forward axis, by the boresight angle, results in true antenna coordinates. In this

case the true antenna coordinates are deflection, range and elevation, with deflection nominally positive-forward, range positive-outward and elevation positive-up. For a right-looking antenna, the boresight angle will be negative. Applying the same rotation would result in a similar deflection, range, negative-elevation system though with negative-elevation oriented positive-down instead of positive-up. To follow convention the negative-elevation is flipped to result in a left-handed deflection, range, elevation system. Thus,

$$\mathbf{R}_{dre}^{xyz} = \begin{cases} \mathbf{R}_{fdl}^{xyz}\mathbf{X}(\theta_b) & \theta_b \geq 0 \\ \mathbf{R}_{fdl}^{xyz}\mathbf{X}(\theta_b)\mathbf{R}_{\text{Right}}^{\text{Left}} & \theta_b < 0 \end{cases} \tag{7.1}$$

The configuration file also stores the desired output grid. This output grid is a rectangular grid in "radar" coordinates, representing range and azimuth positions relative to an SCH reference track. Here the reference track is specified by giving a peg point (latitude, longitude and heading) as well as an altitude. The grid is specified by giving the azimuth and range coordinates of the first pixel, s_0 and r_0, as well as the number of samples in each dimension, N_s and N_r, and the resolution in each dimension, d_s and d_r. Thus the coordinates of a pixel with indices $[u, v]$ can be calculated as

$$\bar{x}_{sr}[u, v] = \begin{bmatrix} s_0 + ud_s \\ r_0 + vd_r \end{bmatrix}. \tag{7.2}$$

Note that, by definition, the range coordinate $x_r[u, v]$ is equivalent to the range of closest approach ρ_0.

A DEM is also specified in the configuration file, which stores ground height above the ellipsoid in a latitude, longitude grid.

At this point, the output files are allocated on disk. There is a file to store the focused imagery, $\sigma(\bar{x})$, one to store the DEM heights, $H(\bar{x})$, one to store a look vector,

$\hat{L}(\bar{x})$, and one to store the accumulated processor gain, $A(\bar{x})$. Each file is in the same output slant-range grid.

7.2.1 Patch Processing

To support very large output grids the processing is split into small patches, where each patch (except for possibly the last one) has some fixed azimuth size, $N_{s,p}$, chosen to maximize computational efficiency. The number of patches is

$$N_p = \left\lceil \frac{N_{s,p}}{N_s} \right\rceil. \tag{7.3}$$

Each patch has an output grid with parameters s_{p0}, N_p, d_s and r_0, N_r, d_r where

$$s_{p0}[k] = s_0 + kN_{s,p}d_s \tag{7.4}$$

and k is the patch index.

At this point a structure to store the processor output, the target grid, is allocated in memory. The target grid contains for each output pixel, $\bar{x}[u,v]$, its complex value, σ, ECEF position, $\bar{x}_{xyz}$, as well as the accumulated look vector, $\hat{L}$, and accumulated gain, A. As part of the allocation the ECEF position is computed for each output pixel. Using the DEM to provide heights, the (s,c,h) cross track coordinate, c, (or equivalently the (s,r,h) coordinate h) is solved for by minimizing the error between h_{calc} and h_{dem} where

$$\begin{aligned}\bar{x}_{\text{calc},llh}[u,v] &= \mathcal{T}_{srh}^{llh}\{\bar{x}_{\text{guess},srh}[u,v]\} \\ &= \mathcal{T}_{xyz}^{llh}\Big\{\mathbf{R}_{x'y'z'}^{xyz}\mathcal{T}_{sch}^{x'y'z'}\left\{\mathcal{T}_{srh}^{sch}\{\bar{x}_{\text{guess},srh}[u,v]\}\right\} + \bar{T}_{x'y'z'}^{xyz}\Big\}\end{aligned} \tag{7.5}$$

and $h_{\text{calc}} = x_{\text{calc},h}$ and h_{dem} is the DEM height interpolated at $(x_{\text{calc},l}, x_{\text{calc},l})$ (the latitude and longitude coordinates) and $x_{\text{guess},h}$ is a solver parameter. This is done for each pixel $[u,v]$.

Given the layout of the GPU kernel as described below, the target grid is transposed and stored in azimuth-major (as opposed to range-major) format. So the index becomes $[v, u]$.

An important step in the patch processing is the calculation of which pulses illuminate the output grid. The idea is to only work with as many pulses as necessary. This calculation is only relevant for stripmap operation. The processor can also be run in spotlight mode, which will always use all available pulses.

The core of this calculation is based upon calculating the slant-range dependent azimuth bounds for a single pulse. For a given pulse and slant-range it is possible to estimate the azimuth positions of the half-power beam edges. For an ideal, un-steered SAR system with azimuth beamwidth θ_b the theoretical resolution can be given as

$$\Delta R = \frac{0.886\lambda}{4\sin\left(\frac{\theta_b}{2}\right)\cos(\theta_{sq})}. \tag{7.6}$$

In this processor the desired processing resolution is specified in the configuration, allowing a variable amount of the beam width to be processed. In this case the beamwidth used by the processor is calculated as

$$\theta_b = 2\arcsin\left(\frac{0.886\lambda}{4\Delta R\cos(\theta_{sq})}\right). \tag{7.7}$$

Using this half-power beamwidth the azimuth offsets from the s position of the beam edges for a given slant range are

$$\Delta s_{n,\pm}[v] = \rho_0[v]\tan\left(\theta_{sq} \pm \frac{\theta_b}{2}\right) \tag{7.8}$$

and the absolute positions can be calculated as

$$E_{n,s,\pm}[v] = p_{n,0,s} + \Delta s_{\pm}[v] \tag{7.9}$$

and converted to grid u indices to give $E_{n,u,\pm}[v]$. Thus for each pulse an array of forward and backward beam edges can be computed, for every slant-range, v.

For a given pulse, this calculation can be used to determine whether or not the output grid contains any of the pulse footprint. Given the potential for non-deterministic squint values, the first and last pulses to illuminate the output grid are determined iteratively. A simple linear search is performed, starting from some reasonable guess and skipping most pulses, for each case until a pulse is found that passes the edge of the patch. The index of the determined pulse is padded to account for any unknown squint, avoiding the need to search every pulse. The result of this search yields $n_{\text{start}}[k]$ and $n_{\text{stop}}[k]$ for the current patch, k.

7.2.2 Range Compression

Using n_{start} and n_{stop}, the pulse data is read from the disk. The stored data may either be $s_{if}[n, t_m]$, for the FMCW case, or $s_{pc}[n, t_m]$, for the pulsed case. In either case the first steps of the NERFFT algorithm are performed here to generate $s_{ner}[n, g]$ where

$$s_{ner,n}(\rho) \approx \frac{1}{\sqrt{2\pi}} \sum_{m=\mu-K}^{\mu+K} \hat{\phi}(\gamma\rho - g) s_{ner}[n, g] \tag{7.10}$$

where $\mu = \lfloor \gamma\rho \rfloor$, in the notation used to describe the NERFFT. The NERFFT parameters γ (the oversampling factor) and K (the interpolator kernel length) are specified in the configuration file.

Normally, for FMCW radars, we have $s_{pc}(n, f) = \mathcal{F}\{s_{if}(n, t)\}$. As a part of the NERFFT interpolation we instead generate

$$s_{ner}[n, g] = \sum_{t_m=-\gamma M/2}^{\gamma M/2-1} e^{-j2\pi g t_m/(\gamma M)} \frac{s_{if}[n, t_m]}{\phi(2\pi t_m/(\gamma M))}. \tag{7.11}$$

The Fourier transform used above in the NERFFT algorithm is symmetric about $t_m = 0$, which allows the window functions to be purely real-valued. To achieve this

process, we first pad with zeros the front and back of the raw data to achieve the desired oversampling, such that

$$s_{tmp}[n,d] = \begin{cases} 0 & 0 < d < a \\ \frac{s_{if}[n,d-a]}{\phi(2\pi(d-a)/(\gamma M))} & a < d < b \\ 0 & b < d \end{cases} \tag{7.12}$$

using the index, d, to represent the γM oversampled data. Then, the front and back halves of the array are swapped so that

$$s_{os}[n,d] = \begin{cases} s_{tmp}[n, d + \frac{\gamma M}{2}] & 0 < d < \frac{\gamma M}{2} \\ s_{tmp}[n, d - \frac{\gamma M}{2}] & \frac{\gamma M}{2} < d < \gamma M \end{cases} \tag{7.13}$$

or

$$s_{os}[n,d] = \begin{cases} \frac{s_{if}[n,d+\frac{M}{2}]}{\phi(2\pi(d+\frac{M}{2})/(\gamma M))} & 0 < d < \frac{M}{2} \\ 0 & \frac{M}{2} < d < \frac{\gamma M}{2} \\ \frac{s_{if}[n,d+\frac{M}{2}]}{\phi(2\pi(d+\frac{M}{2})/(\gamma M))} & \frac{\gamma M}{2} < d < \frac{\gamma M+M}{2} \\ 0 & \frac{\gamma M+M}{2} < d < \gamma M \end{cases} . \tag{7.14}$$

Using a standard, non-symmetric, FFT as provided by most software packages, of the form

$$X[l] = \sum_{k=0}^{K} e^{-j2\pi lk/K} x[k] \tag{7.15}$$

gives the desired result

$$s_{ner}[n,g] = \sum_{k=0}^{\gamma M} e^{-j2\pi \frac{gd}{\gamma M}} s_{os}[n,d]. \tag{7.16}$$

Due to the nature of the FMCW system, this result contains redundant positive and negative frequency parts. The positive frequency part is chosen so that the index g is proportional to range, as in

$$s_{ner}[n,g] = s_{ner}[n,g] \qquad \forall g \leq \frac{\gamma M}{2}. \tag{7.17}$$

The pulsed radar data is stored in the range-compressed form, but must be modified to work with the NERFFT. The final step in pulse-compressing the pulsed radar data is an IFFT, which can be modified to work with the NERFFT.

First, the pulse-compressed data is put back into the frequency-domain, padding the number of samples to an efficient number for computation with the IFFT, M. Here,

$$s_{tmp}[n,f] = \frac{\mathcal{F}\{s_{pc}[n,m]\}}{\phi(2\pi(f+\frac{M}{2})/(\gamma M))} \tag{7.18}$$

and is again shifted such that

$$s_{os}[n,f] = \begin{cases} s_{tmp}[n,d] & 0 < d < \frac{M}{2} \\ 0 & \frac{M}{2} < d < \gamma M - \frac{M}{2} \\ s_{tmp}[n,d+\frac{M}{2}] & \gamma M - \frac{M}{2} < d < \gamma M \end{cases}. \tag{7.19}$$

Finally, an inverse FFT can be applied to give the NERFFT pulse-compressed data

$$s_{ner}[n,g] = \mathcal{F}^{-1}\{s_{os}[n,d]\} \tag{7.20}$$

where again the g index is proportional to range. The Fourier transforms for this step are computed on the GPU, resulting in the output $s_{ner}[n,l]$ located in GPU global memory.

For convenience, all pulse related data structures are sampled to match the size calculated for the current patch. For the following steps, the pulse index, n, is assumed to start at $n_{\text{start}}[k]$ and end at $n_{\text{stop}}[k]$.

7.2.3 Azimuth Compression Module

Here, all of the configuration and data structures for the patch are passed to the azimuth compression module. The azimuth compression module starts by loading and compiling the GPU kernels. Compiling these kernels at runtime allows modifications to the kernel before the compilation, which can improve performance. Space is allocated in the GPU memory for the target grid structure, the radar state structure and arrays to hold the pulse footprint boundaries.

If using the global pulse boundary setting, the azimuth offsets, $\Delta s_{n,\pm}[v]$, are calculated here using the provided slant-range dependent squint angle, $\theta_{sq}[v]$.

The loop over all pulses, $n \in [n_{\text{start}} : n_{\text{stop}}]$, begins by checking the radar state flags to see if n is an invalid pulse. If the pulse is valid, the processing continues. The azimuth offsets for the current pulse, $\Delta s_{n,\pm}[v]$, are calculated here when using the local pulse boundary setting. Using the offsets, $\Delta s_{n,\pm}[v]$, the boundary positions for the pulse n are again calculated as

$$E_{n,s,\pm}[v] = p_{n,0,s} + \Delta s_{\pm}[v] \tag{7.21}$$

and copied to the GPU.

The optional antenna pattern is (re)calculated (if the radar state indicates a change in the electronic steering angle) and copied to the GPU. If the boundary positions, $E_{n,s,\pm}[v]$, indicate the pulse illuminates any of the scene, the GPU kernel is executed to perform the actual accumulation. Any additional relevant configuration and/or memory pointers are passed to the GPU kernel as arguments.

7.2.4 GPU Kernel

The kernel is arranged such that each block processes one azimuth line (a constant range value), such that $i_{\text{block}} = v$, and each thread in the block processes one pixel in that line. Since the block size is fixed, some threads may process more than one

pixel, or no pixels at all, depending on the azimuth bounds for that range. Initially, $u = i_{\text{thread}} + E_{n,u,-}[v]$. The radar state entry for pulse n is loaded into the shared memory for the block.

To fully utilize the GPU memory bandwidth, memory accesses should be sequential. Since individual blocks process pixels from a single range line (constant v), the target grid has been arranged in azimuth-major format, so sequential memory locations are adjacent azimuth pixels (u).

Additionally, a subset of the pulse-compressed data, s_{ner}, is loaded into the shared memory. For a given block, at range $\rho_0[v] = r_0 + vd_r$, there will be a limited range of pulse-compressed values needed. This is estimated by using the pulse boundaries calculated from the reference track, which will correspond to a range of

$$\rho_{b,\pm} = \sqrt{\rho_0^2 + (p_{n,0,s} - E_{n,s,\pm}[v])^2}. \tag{7.22}$$

The range at closest approach, $\rho_0[v]$, will be the smallest possible range, except for large squint angles where it is not within the beam footprint. The range, $g_\pm$, bounds for s_{ner} are chosen from the minimum and maximum of the above three ranges appropriately and are padded to account for non-ideal motion of the platform (since the true range will be calculated from the platform position, and not the reference track). Thus, the subset $s_{ner}[n, g_- : g_+]$ is loaded into shared memory.

Each pixel/thread calculation starts by loading the values from the target grid structure for the current pixel $[v, u]$. This contains the target position, $\bar{x}[v, u]$, current value $\sigma(\bar{x}[v, u])$, current look vector, $\bar{L}(\bar{x}[v, u])$, and current gain, $A(\bar{x}[v, u])$. The real distance is calculated as

$$\rho(\bar{x}, \bar{p}_{n,0}) = \|\bar{x} - \bar{p}_{n,0}\| \tag{7.23}$$

and the look vector as

$$\hat{L}_n(\bar{x}) = \begin{bmatrix} (x_x - p_{n,0,x})/\rho(\bar{x}, \bar{p}_{n,0}) \\ (x_y - p_{n,0,y})/\rho(\bar{x}, \bar{p}_{n,0}) \\ (x_z - p_{n,0,z})/\rho(\bar{x}, \bar{p}_{n,0}) \end{bmatrix}. \tag{7.24}$$

Without an antenna pattern, the gain value is set to 1. Otherwise the antenna pattern gain is calculated. The antenna pattern is stored in a lookup table using the GPU texture cache and is indexed using normalized deflection and elevation coordinates. The normalized coordinates of the pixel position in the antenna dre system is calculated as

$$\hat{x}_{dre} = \frac{1}{\rho(\bar{x}, \bar{p}_{n,0})} \mathbf{R}^{dre}_{xyz}(\bar{x}_{xyz} - \bar{p}_{n,0,xyz}) = \mathbf{R}^{dre}_{xyz} \hat{L}_n(\bar{x}). \tag{7.25}$$

The values x_d and x_e are used to lookup the antenna gain, $G_n(\bar{x})$. The total gain for the pixel also takes into account the range loss, which may have an additional normalization parameter depending on the range-compression method

$$A_n(\bar{x}) = G_n(\bar{x}) \frac{\rho^2_{\text{norm}}}{\rho^2}. \tag{7.26}$$

The phase correction is calculated as

$$\phi_{\text{ref}}(\bar{x}, n) = -j2\pi f_0 \frac{2\rho(\bar{x}, \bar{p}_{n,0})}{c} \tag{7.27}$$

for pulsed radars and

$$\phi_{\text{ref}}(\bar{x}, n) = (\text{UD}) j2\pi f_0 \frac{2\rho(\bar{x}, \bar{p}_{n,0})}{c} - j\pi K_r \left(\frac{2\rho(\bar{x}, \bar{p}_{n,0})}{c} \right)^2 \tag{7.28}$$

for FMCW radars, where UD $= -1$ for an up-chirp and UD $= 1$ for a down-chirp.

The range-compressed interpolation starts by converting $\rho(\bar{x}, \bar{p}_{n,0})$ into a range index

$$\rho_g(\bar{x}, \bar{p}_{n,0}) = \frac{\rho(\bar{x}, \bar{p}_{n,0}) - r_0}{\Delta r}. \tag{7.29}$$

For the pulsed case, the interpolation is completed as

$$s_{ner}[n, \rho_g] = e^{-j\pi\rho_g} \sum_{g=\mu-K}^{\mu+K} \frac{\gamma}{G} \frac{1}{\pi} \frac{\sinh\left(\alpha\sqrt{K^2 - (\gamma\rho_g - g)^2}\right)}{\sqrt{K^2 - (\gamma\rho_g - g)^2}} s_{ner}[n, g] \tag{7.30}$$

where $\mu = \lfloor \gamma\rho_m \rfloor$, and in the FMCW case as

$$s_{ner}[n, \rho_g] = \sum_{g=\mu-K}^{\mu+K} \frac{1}{\pi} \frac{\sinh\left(\alpha\sqrt{K^2 - (\gamma\rho_g - g)^2}\right)}{\sqrt{K^2 - (\gamma\rho_g - g)^2}} s_{ner}[n, g]. \tag{7.31}$$

The calculated values are now accumulated into the target grid as

$$\begin{aligned}
\sigma(\bar{x}) &= \sigma(\bar{x}) + s_{ner}[n, \rho_g] e^{j\phi_{\text{ref}}(\bar{x}, n)} (A_n(\bar{x}))^{(\text{MD})1} \\
\bar{L}(\bar{x}) &= \bar{L}(\bar{x}) + \bar{L}_n(\bar{x}) (A_n(\bar{x}))^{(\text{MD})2} \\
A(\bar{x}) &= A(\bar{x}) + (A_n(\bar{x}))^{(\text{MD})2}
\end{aligned} \tag{7.32}$$

where the gain may either be multiplied or divided, according to the configuration option MD. The current pixel values used in the accumulation sum (the first terms) are those read at the start of the pixel calculation. If there are more pixels in $[E_{n,u,-}[v], E_{n,u,+}[v]]$ than there are threads in the block then each thread loops until all pixels are complete.

7.2.5 Grid Finalization

When the GPU kernel has completed all pixels for the current pulse, the loop over pulses, n, iterates until all pulses have been processed. Before the patch is completed the target grid is finalized as

$$\sigma(\bar{x}) = \frac{\sigma(\bar{x})}{A(\bar{x})}$$
$$\hat{L}(\bar{x}) = \frac{\bar{L}(\bar{x})}{\left\|\bar{L}(\bar{x})\right\|} \tag{7.33}$$

Finally, the completed patch is returned from the azimuth compression module.

The completed patch is written to disk and the loop over patches is continued. After the last patch is complete there are no other processing steps, the completed image has been fully compressed and written.

7.3 Summary

A detailed look into each step of the processor was given in this chapter. This is the culmination of the work described in previous chapters, showing how the theoretical models and implementation details come together to form a working time-domain backprojection SAR processor. Since the layout of this chapter is as the program is written, it also provides a useful reference when reading the written program code. Using the processor as described here, the next chapter presents results obtained for the relevant radar systems.

CHAPTER 8

RESULTS

Using this processor, preliminary results have been obtained with the two UMass systems and the UAVSAR instrument. Additionally, simulated data were processed to help evaluate the quality and accuracy of the processor itself.

Corner reflectors, due to their well behaved impulse response, are commonly used as point targets for radar system calibration. The system impulse response is often quantified by several measures, as described by Cumming [7]. The first is resolution, or impulse response width, IRW, defined as the half-power beamwidth of the main lobe of the impulse response. This can be measured in any distance unit. Here meters are used. The peak sidelobe ratio, PSLR, measures the power of the first peak outside of the main lobe relative to the main lobe's peak. It is measured in dB. The integrated sidelobe ratio, ISLR, is a measure of the power in the main lobe relative to the entire power in the side lobes, defined as

$$\text{ISLR} = 10\log_{10}\left(\frac{P_{total} - P_{main}}{P_{main}}\right). \tag{8.1}$$

In this case the width of the main lobe is defined as the null-to-null beamwidth, or $\frac{\text{IRW}}{0.886}$. The ISLR may either be measured in a single dimension (here we use azimuth and range), or in two dimensions where it is referred to as ISLR_{2D}. All of these quantities may be measured and used as performance metrics for the processor and/or radar system. While the area surrounding a corner reflector will not be empty in practice, well chosen sites will have little impact due to the extraordinarily large reflectivity of the reflector. It should be noted that the application of window functions

to the radar data, in the range and/or azimuth directions, are often used to trade off performance between these metrics.

8.1 Simulated Data

Using a simulator developed at the Jet Propulsion Laboratory, SAR data was generated with radar and flight parameters similar to a typical UAVSAR deployment. The simulation contains a number of corner reflectors on the ellipsoid surface (i.e. there is no DEM). Figure 8.1 shows the target reflectivity image as computed by the processor, for the simulated data. Two corner reflectors can be seen in the top of the image, as well as the azimuth sidelobes for one reflector outside of the image. Note that in such a small scene as this, there are many pulses which cover all or most of the scene. The point target response of the nearest reflector is shown in Figure 8.2 and the measured impulse response characteristics in Table 8.1.

Table 8.1: Simulated point target impulse response.

Metric	Value
Range IRW (m)	2.30
Range PSLR (dB)	36.04
Range ISLR (dB)	-25.9
Azimuth IRW (m)	1.09
Azimuth PSLR (dB)	18.47
Azimuth ISLR (dB)	-16.06
ISLR_{2D} (dB)	-15.6

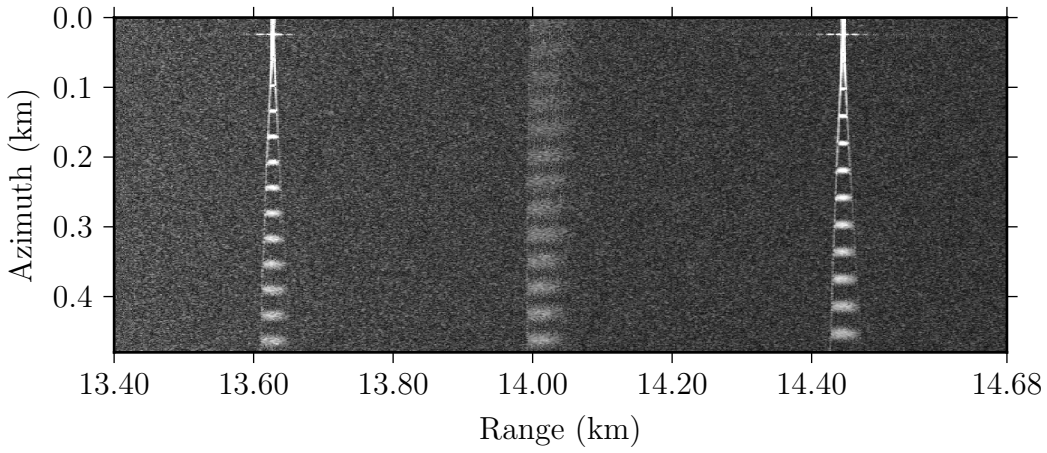


Figure 8.1: Slant-range reflectivity output for simulated corner reflector scene.

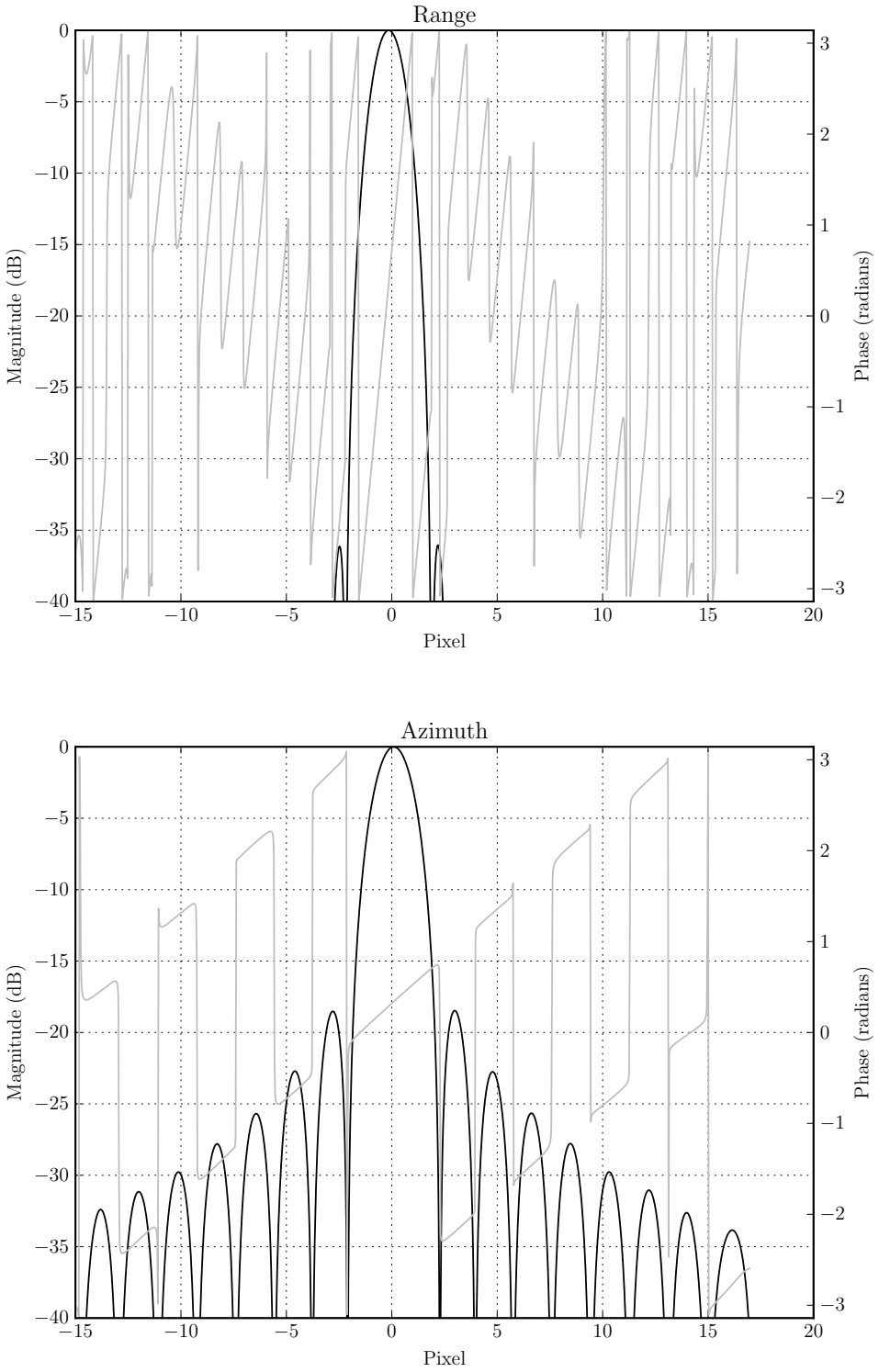


Figure 8.2: Oversampled point target response in simulated image. Note the well behaved magnitude and phase responses.

8.2 UAVSAR Data

The UAVSAR instrument developed and operated by NASA's Jet Propulsion Laboratory has a long operating history. While typically processed with frequency-domain algorithms, the results here show that time-domain backprojection may be a viable alternative, especially for scenes with high motion variance. The reflectivity and point target response is shown for a single flight over a calibration site in California. While the results are generally quite good, there is clearly still some error causing the smearing of the point target response, mainly in the azimuth direction.

Table 8.2: UAVSAR measured point target impulse response.

Metric	Value
Range IRW (m)	2.45
Range PSLR (dB)	16.90
Range ISLR (dB)	-18.0
Azimuth IRW (m)	3.17
Azimuth PSLR (dB)	2.64
Azimuth ISLR (dB)	-8.2
ISLR_{2D} (dB)	-7.6

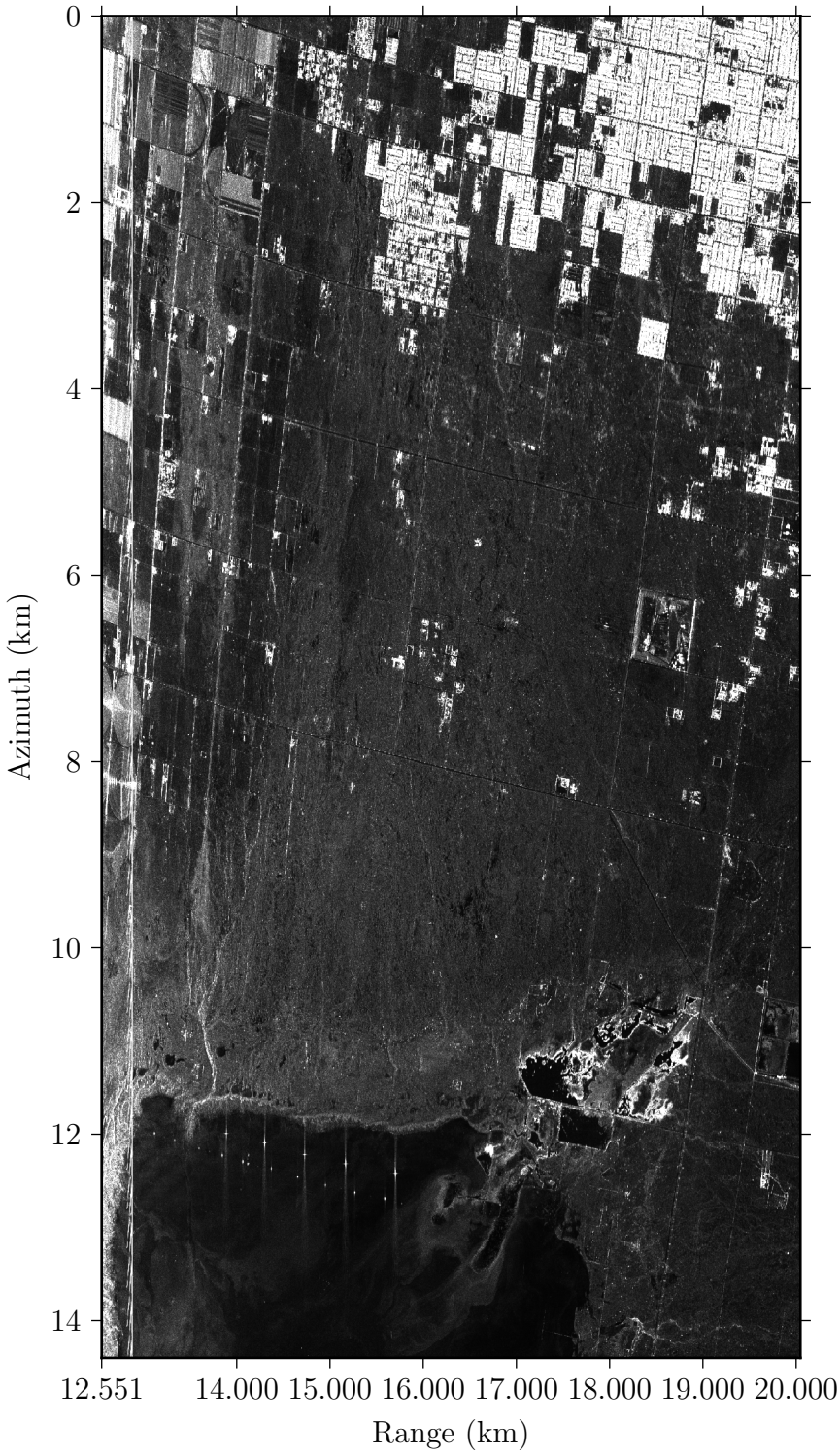


Figure 8.3: Slant-range reflectivity output for UAVSAR scene. Note the corner reflectors in the bottom left of the image.

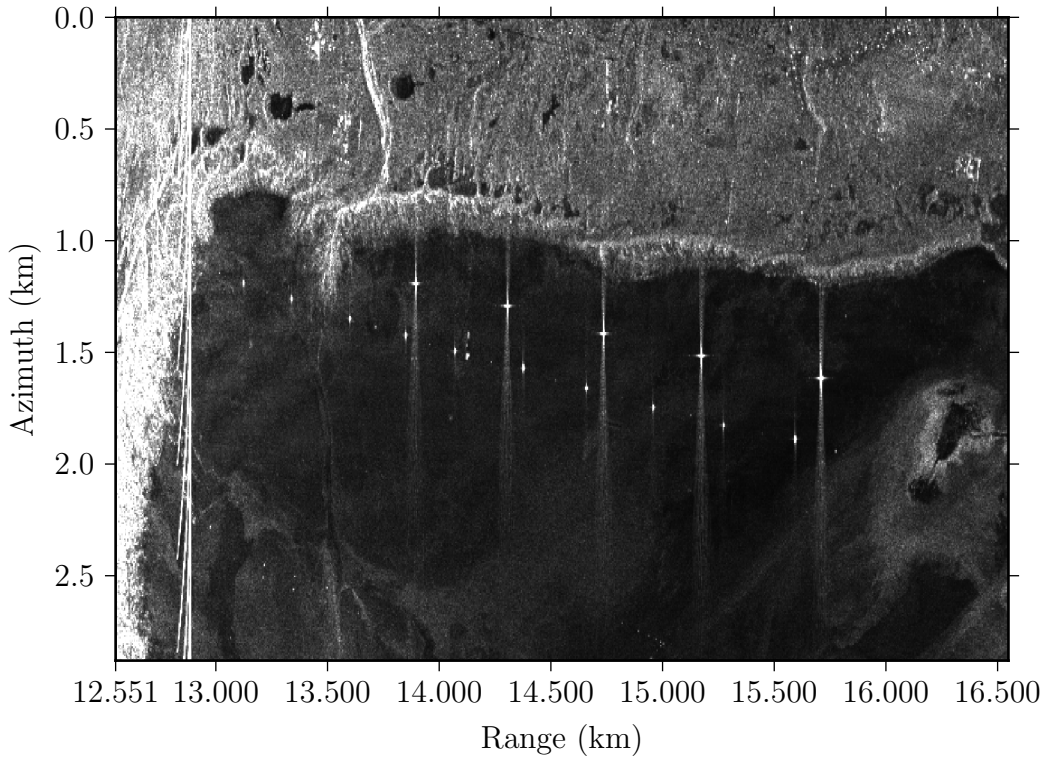


Figure 8.4: A closeup of the corner reflectors in the UAVSAR scene.

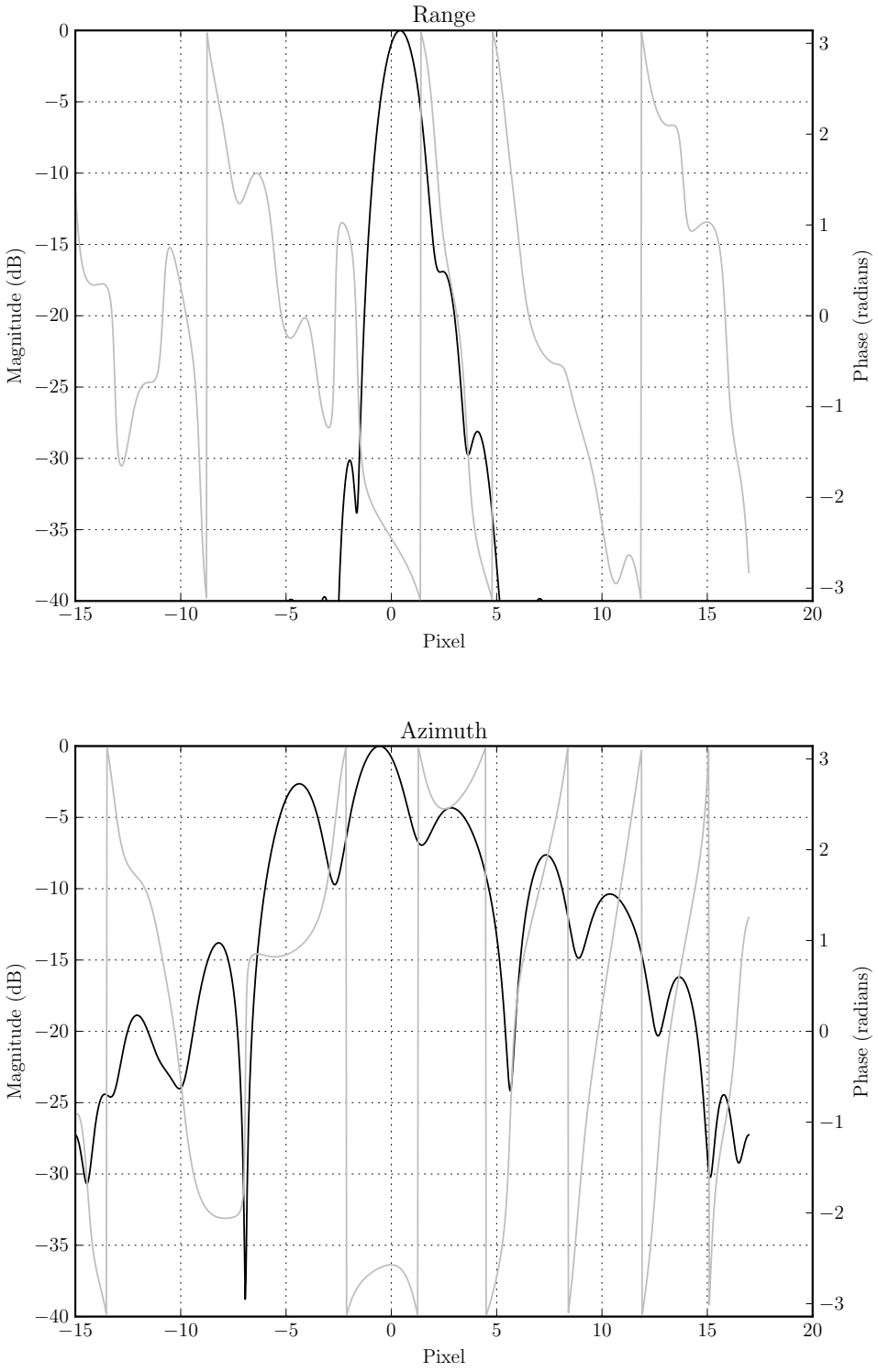


Figure 8.5: Oversampled response for corner reflector in UAVSAR scene.

8.3 UMass Ka-band Data

The UMass Ka-band radar has been deployed in many configurations, including not only as a SAR instrument. Here results are presented for the dual S- and Ka-band deployment, for which both systems were deployed simultaneously on the same platform as cross-track interferometers. In addition to pure reflectivity, each single interferometer is able to measure interferometric phase and estimate precise scatter height. A coherence image estimated for the two channels provides the interferometric phase, which can be converted to a scatter height as described in Chapter 4. The scatterer height estimate is shown in Figure 8.9. Using the DEM, in the same output grid, and the estimated scatterer heights, a "final" DEM can be derived as the sum of the two. This final product is a measure of the absolute scatter height, as estimated by the interferometer. Figure 8.7 and Figure 8.11 show these final heights. Before the addition is performed, the scatterer heights, $h[u, v]$, are weighted by the coherence magnitude $|\gamma[u, v]|$ such that

$$h_{\text{final}}[u, v] = h[u, v]|\gamma[u, v]|. \tag{8.2}$$

Since the coherence magnitude ranges from zero to one, this weighting is a convenient way to reduce the effects of noisy pixels on the final heights (for example allowing reasonable values in shadow regions).

It is clear to see the degradation caused by the residual motion errors (measured incorrectly or unmeasured) in the results. It is most noticeable as the near horizontal banding that appears in both the reflectivity and phase based results. The banding is most likely caused by rapid changes in platform motion which are not precisely measured. Due to its small wavelength and small beamwidth the Ka-band system is particularly prone to these types of errors.

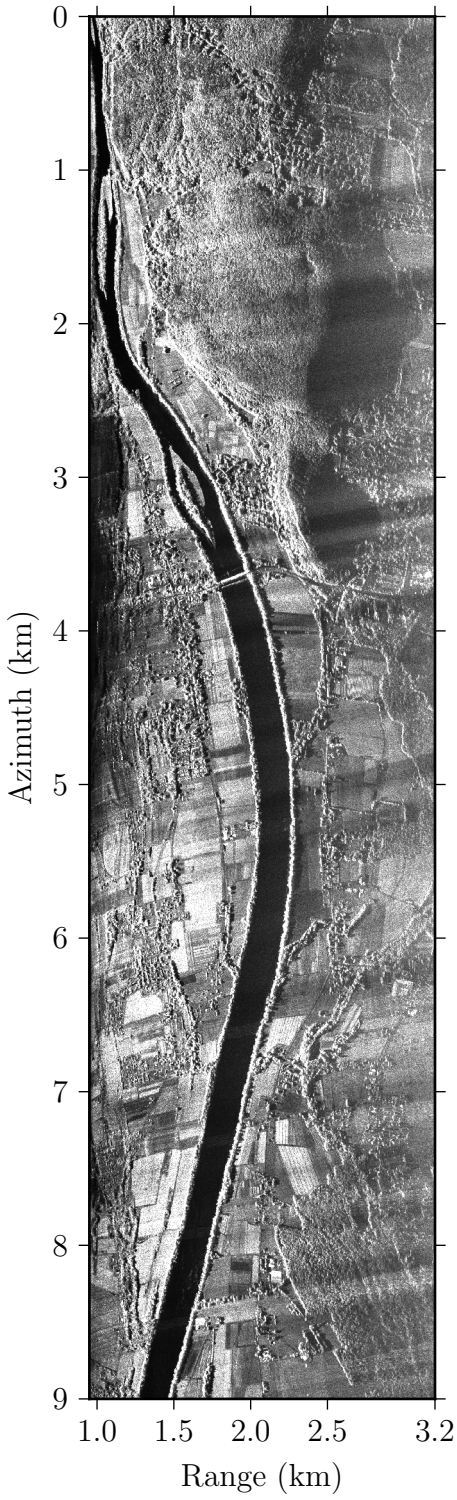


Figure 8.6: Ka-band reflectivity image. Note the magnitude patterns caused by incorrect motion.

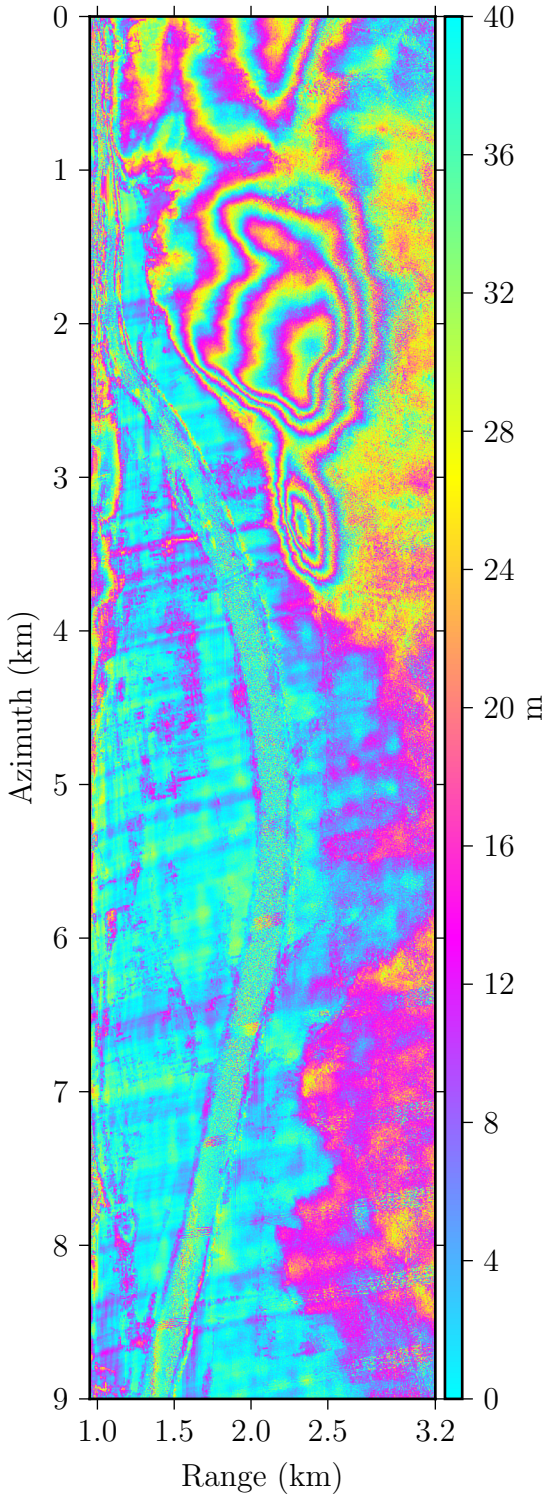


Figure 8.7: Final DEM using Ka-band interferometric height estimates. Note that the height estimates are weighted by the coherence magnitude.

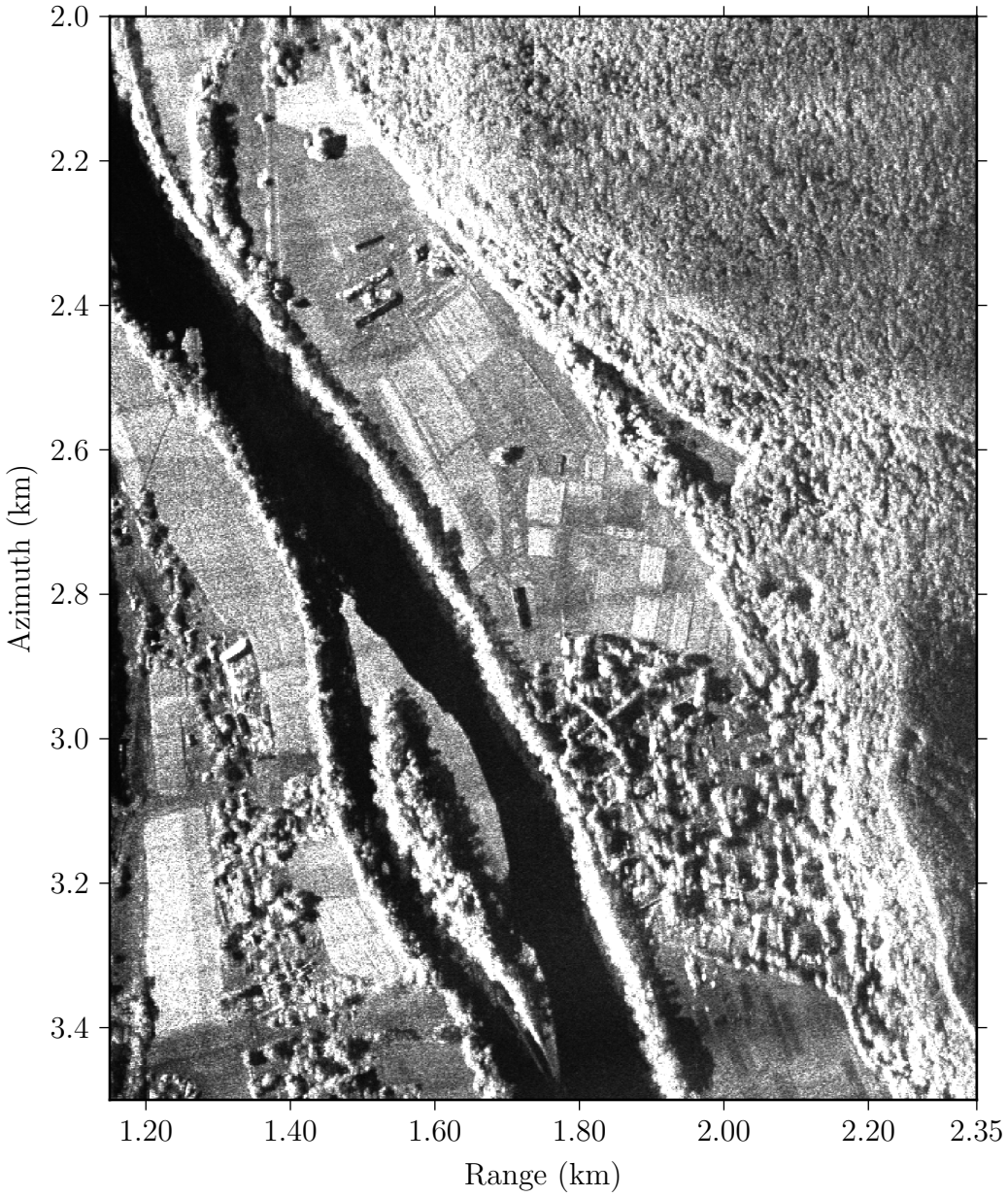


Figure 8.8: Ka-band reflectivity detail image.

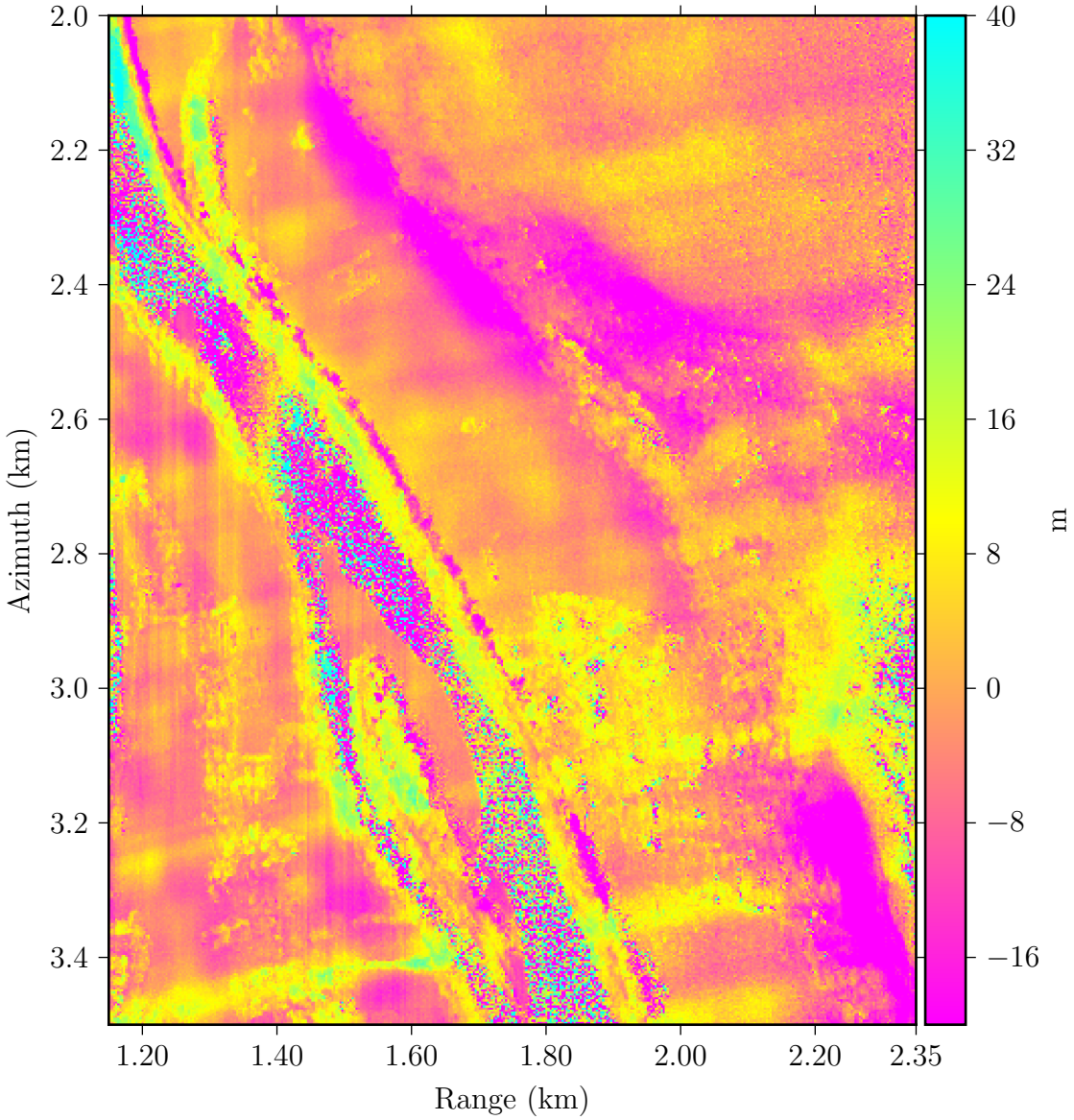


Figure 8.9: Ka-band raw differential height estimate.

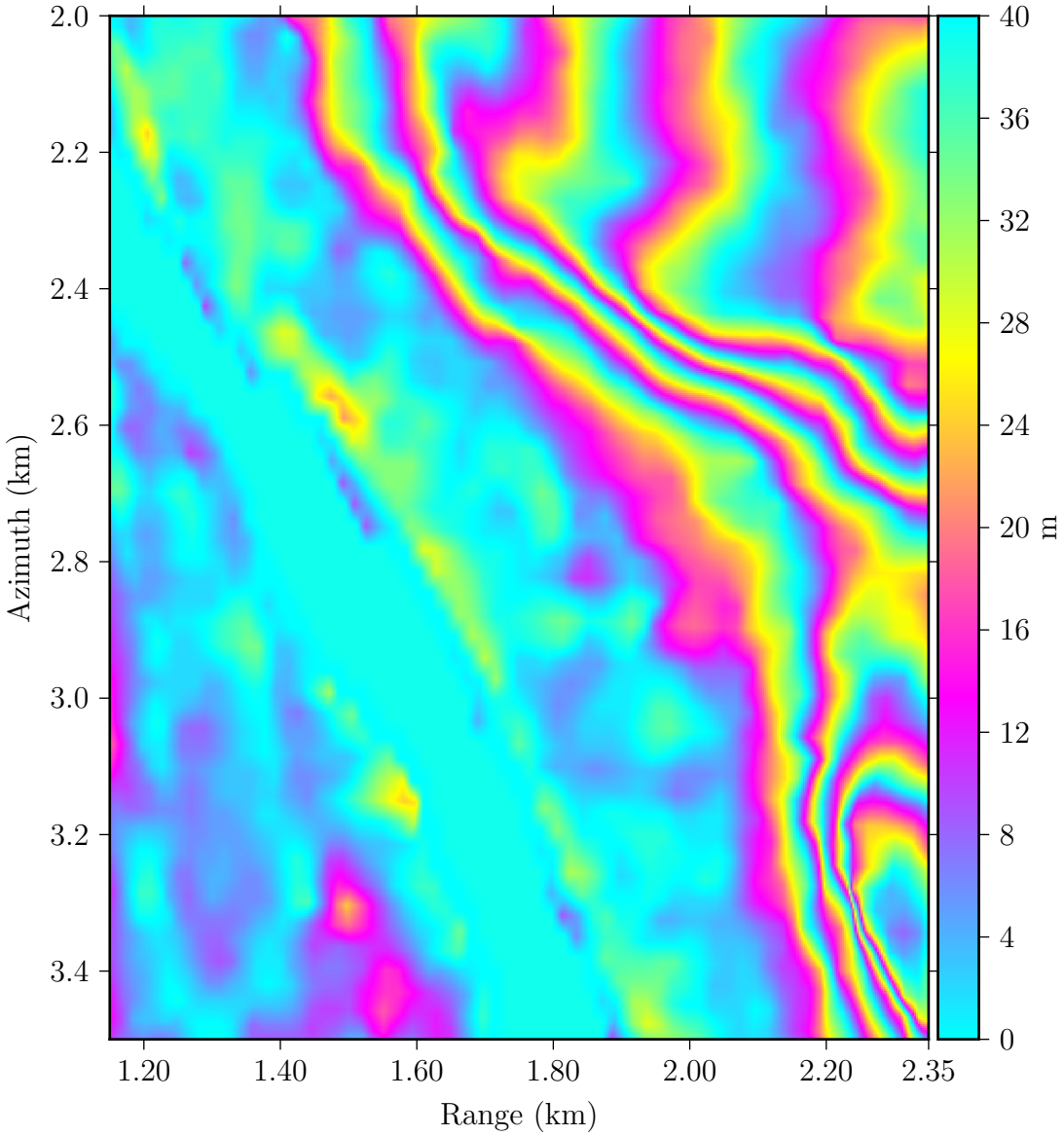


Figure 8.10: Reference DEM used for processing.

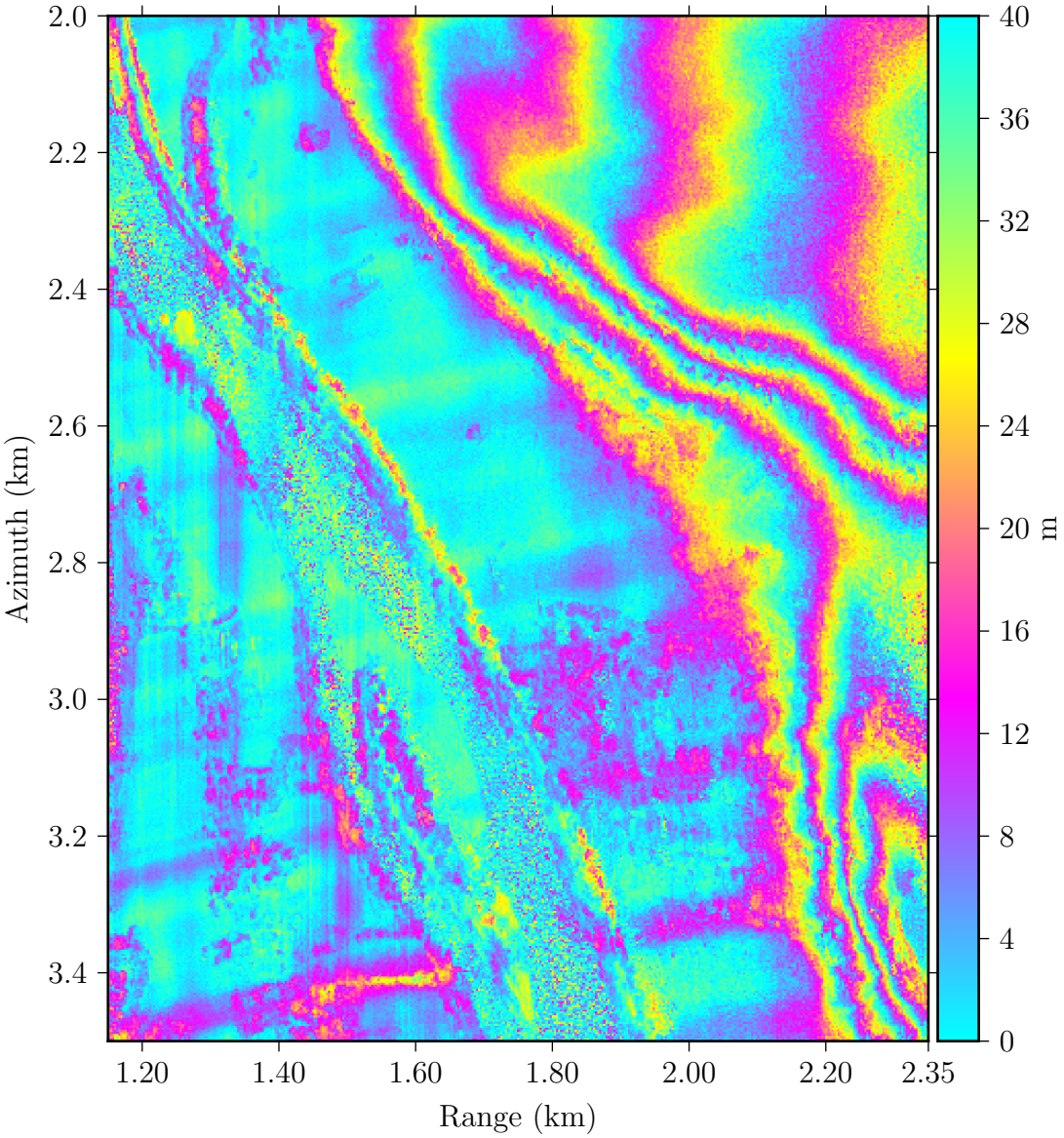


Figure 8.11: Final Ka-band DEM detail image.

8.4 UMass S-band Data

The UMass S-band system, as deployed in conjunction with the Ka-band system, can also be processed in the same way. In fact, the nature of the processor makes it easy to process the two images to the same output grid, allowing pixel by pixel comparison. The results in the following pictures were processed in the same manner as the Ka-band system, on the same grid, for the same data acquisition flight. As would be expected the results look fairly similar.

Overall the S-band system has more consistent results than the Ka-band system, as it is less prone to the same types of motion errors. Due to the much larger aperture, however, the S-band system does suffer from degraded azimuth resolution due to lower frequency motion errors. Though the two systems should theoretically have the same resolution, this affect causes the S-band system to have lower azimuth resolution, as seen. Some radar hardware issues can be seen in the imagery, most noticeably the phase response in noisy areas (which is expected to look more noisy).

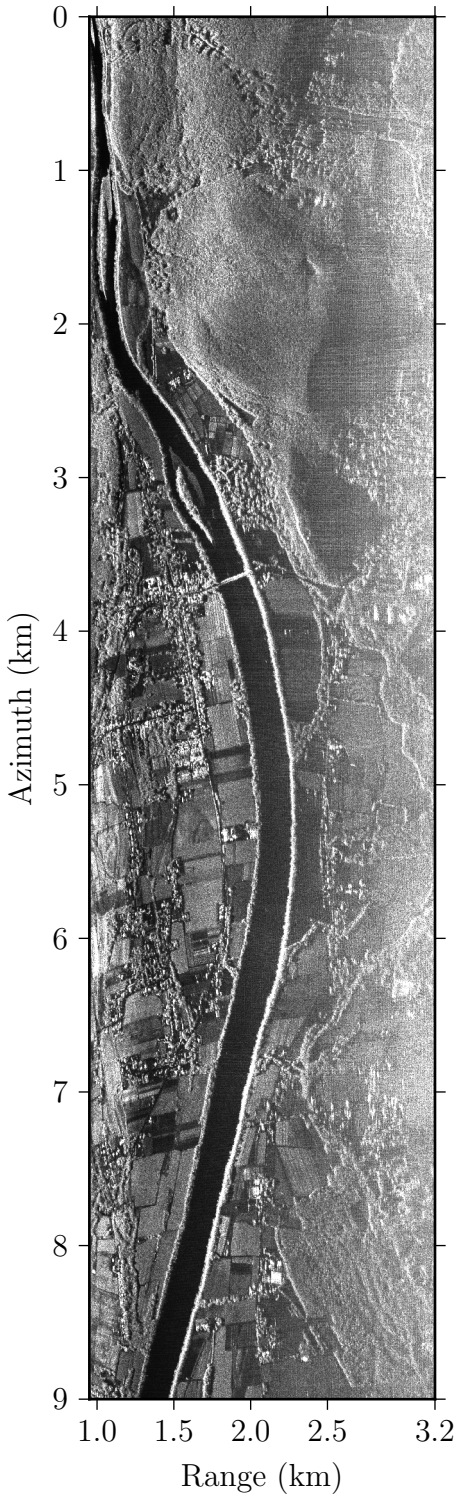


Figure 8.12: S-band reflectivity image.

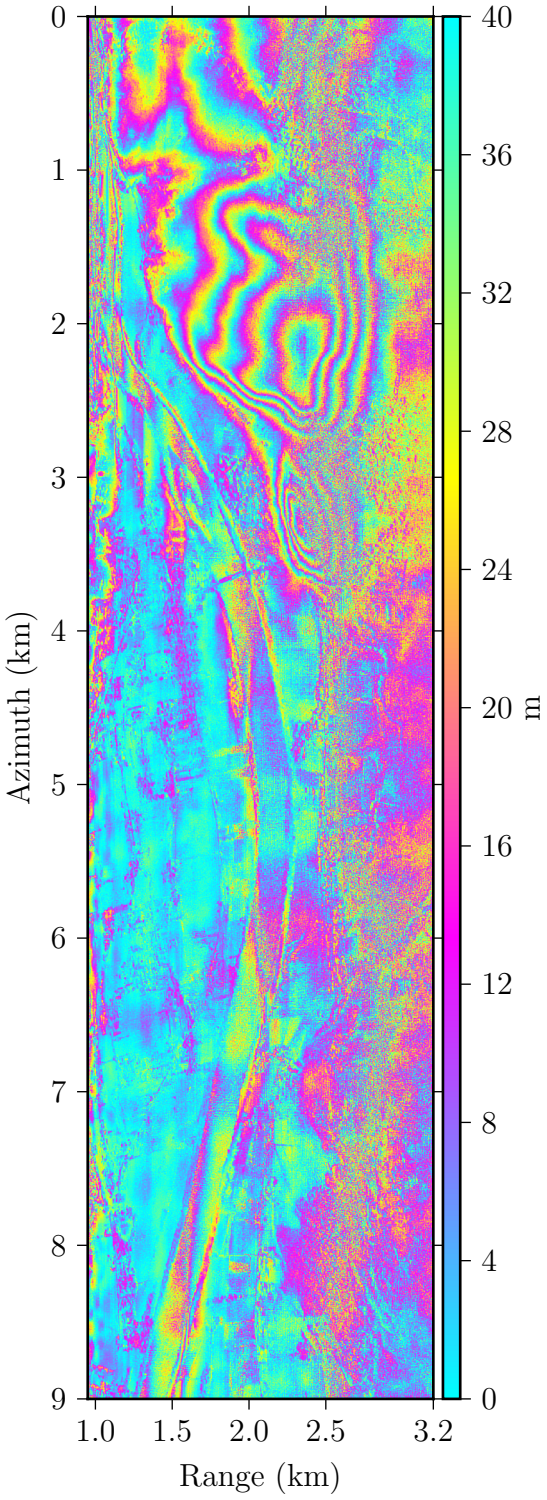


Figure 8.13: Final DEM using S-band interferometric height estimates. Note that the height estimates are weighted by the coherence magnitude.

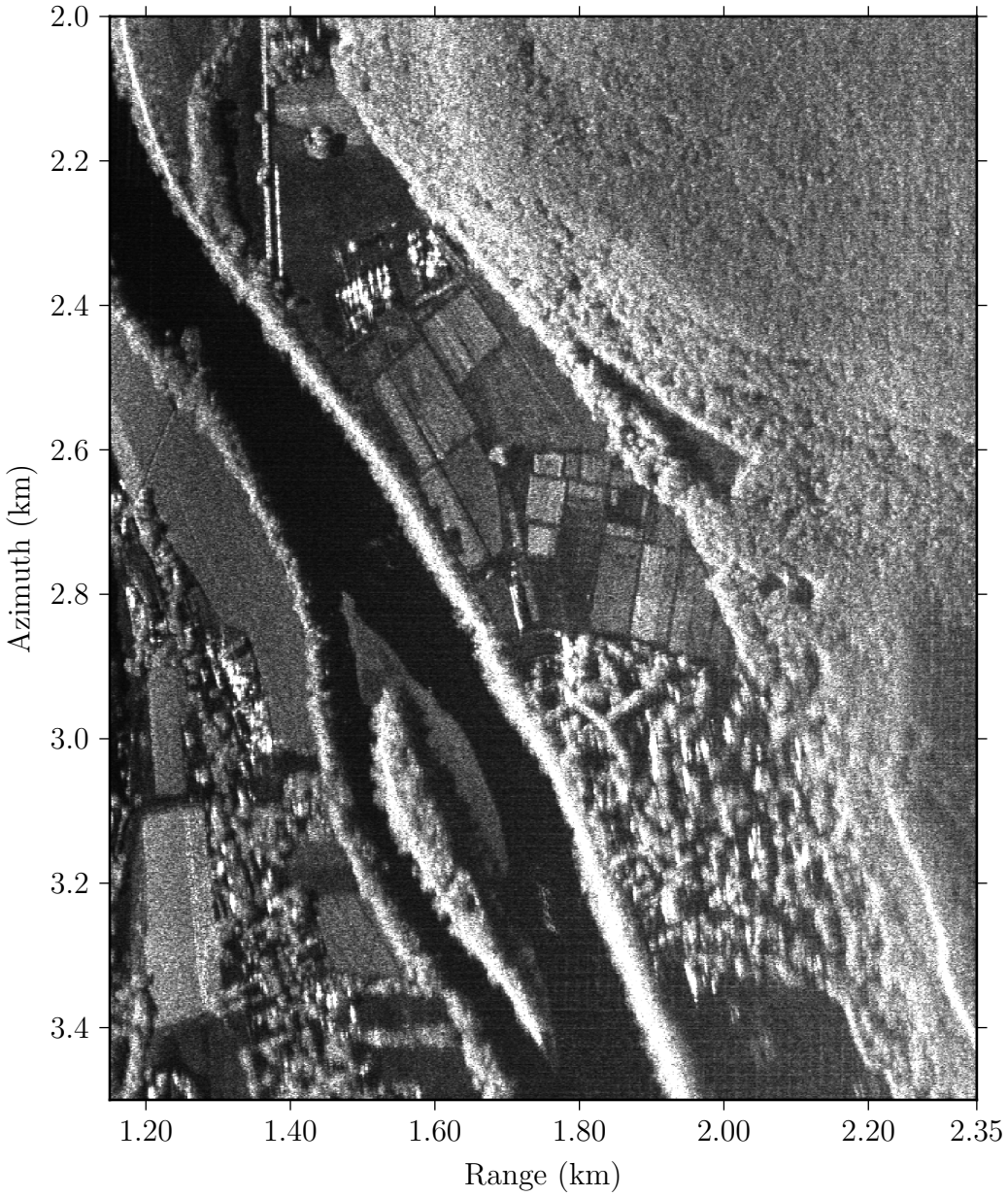


Figure 8.14: S-band reflectivity detail image.

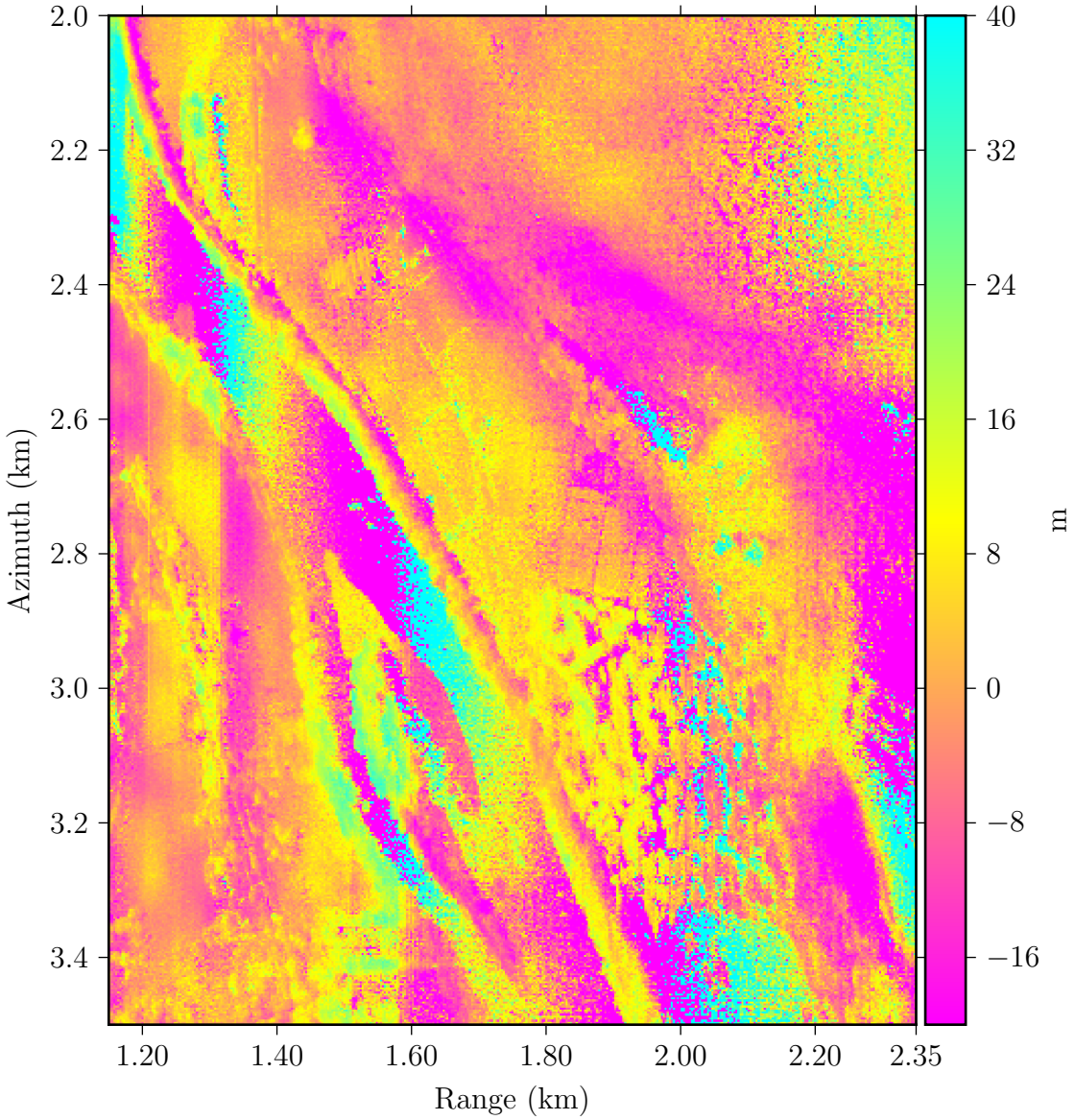


Figure 8.15: S-band raw differential height estimate.

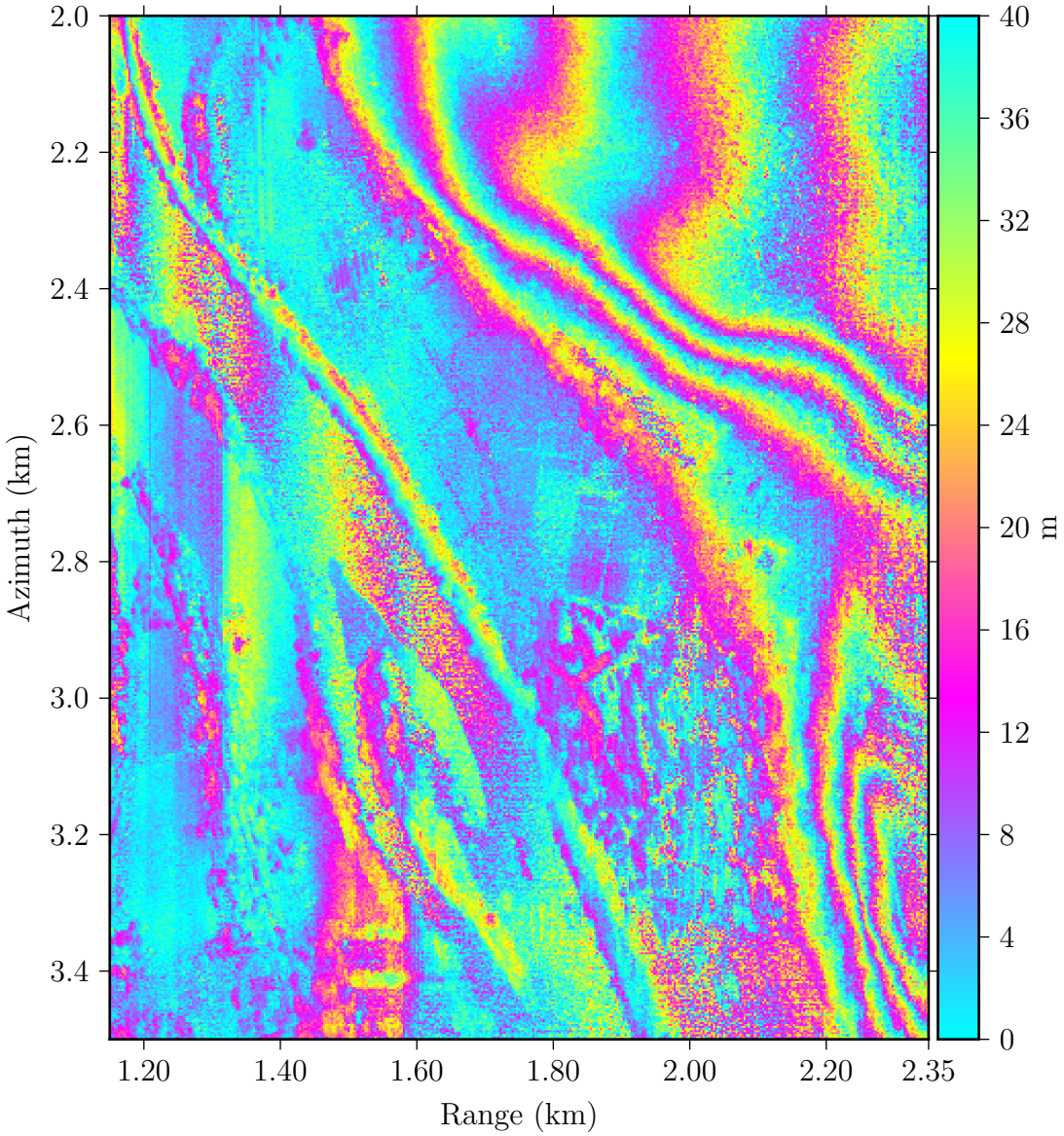


Figure 8.16: Final S-band DEM detail image.

8.5 Combined S and Ka-band Data

The unique nature of the dual-frequency UMass system allows direct comparison of measured data for each frequency. Figure 8.17 shows the difference in height between the Ka-band and S-band measured scatterer heights. Due to the order of magnitude difference in system wavelengths it is expected for the two systems to have much different scattering statistics. One interesting observation is the generally positive difference, especially over individual trees and forests (for example in the upper right of the image, or the tree at approximately 2.2km azimuth and 1.5km range). This is consistent with the hypothesis that the Ka-band system will scatter more directly off tree canopies, while the S-band system would not. There is clearly potential for further study of the dual-frequency response.

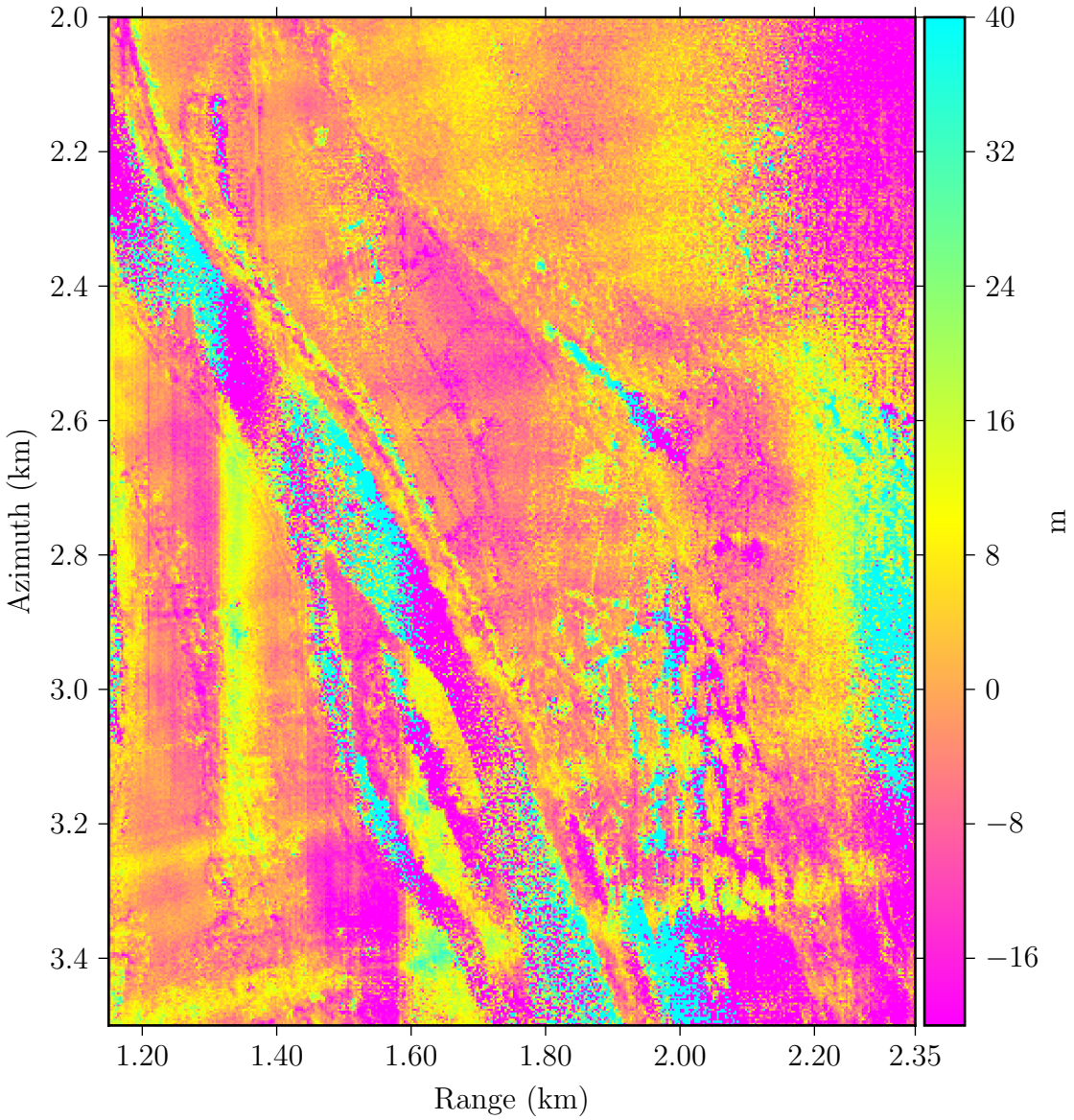


Figure 8.17: Height difference measured between Ka and S-band images. Note the general positive bias over forested areas.

CHAPTER 9

CONCLUSION

The majority of this work entailed the development of a time-domain backprojection processor, designed to use GPU hardware to improve runtime performance, and to work with a wide variety of airborne radar systems. The goal of the processor is to enable accurate processing in the face of non-linear aircraft motion, which is difficult to avoid, especially in low altitude aircraft. Time-domain backprojection makes only minor assumptions about aircraft motion, much of which is correctable as described, making it particularly suitable for the application. To achieve good results, the processor must have good knowledge of the motion of the aircraft, which is not always easy to obtain. The motion data may be estimated and/or refined, using the Doppler information in the recorded radar data.

A critical step of the backprojection algorithm is the interpolation of the pulse-compressed data. Here, the NERFFT method is used, for both FMCW and pulsed systems. This method is able to provide high accuracy with limited computational cost.

Another important aspect of the algorithm is the substantial computational complexity. Though much less than that of pure time-domain correlation, the complexity is still significant when compared to traditional frequency-domain approaches. While this processor generally stresses accuracy, flexibility and usability the implementation medium was chosen to provide high performance.

Interferometry is an important aspect of the relevant radar systems. There has been limited work using backprojection processed imagery for interferometry and its

ultimate suitability may be uncertain. Theoretically, it is shown to provide different performance over traditional methods, which generally use frequency-domain processing algorithms. Preliminary interferometric results have been shown using the backprojection processor and show promise.

9.1 Contributions

This main contributions of this work revolve around the development and preliminary results of a time-domain backprojection processor.

- Development of a time-domain backprojection processor with the following properties:
  - Ability to process pulsed and FMCW data, from a wide range of system parameters.
  - Designed to generate highly accurate focused SAR imagery and accompanying data products.
  - Full compensation of antenna pattern and variable aperture length gain.
  - Support for very large radar scenes.
  - Core processing run on GPU hardware to improve performance.
  - Flexibility to work on desktop PCs and computing clusters.
- Development of a preprocessor for the UMass radar systems.
  - Combines GPS, INU and radar timing data into one structure to be used by the processor.
  - Properly addresses antenna position offsets on platform using INU data.
- Development of accompanying tools for interferometry and data manipulation.
- Evaluation of processor performance with simulated and collected data sets.

- Preliminary results performing interferometry with backprojection processed SAR imagery.
- Preliminary interferometric results using the UMass dual-frequency S and Ka-band radar system, comparing estimated scatterer height values.

9.2 Future Work

Further improvements may be made to the processor. One option to correct for unknown platform motion is to perform some type of autofocusing during data processing. This is commonly done with frequency-domain algorithms and has the potential to be extended to time-domain backprojection. The unknown motion may also be corrected before processing. This can be done using estimation procedures that rely on Doppler information in the radar data. This is in part what is currently done with the UMass radar systems, but is very basic. Another option is to improve the motion measurement system itself, to improve the initial platform motion estimation. The UMass systems are in the process of upgrading the GPS and INU on the aircraft with this goal in mind. The UAVSAR system already has very sophisticated motion measurement capabilities, which is at work enabling the high quality results. While good, the results are not perfect and it may be worth investigating the cause of any errors or reduced performance, in order to come up with the best method for mitigating them. Likely, autofocus methods would improve the image quality for many types of error sources.

While the processor is basically complete, and able to stand on its own, there are still some additional improvements which may be made. Some features which could be implemented are a full 2D antenna pattern (instead of two 1D patterns), ability to process a squinted output geometry, to improve performance for highly squinted data sets, improved computational performance and modularity and better support for spotlight data.

With little previous work found on interferometry with backprojection, further investigation is warranted. The preliminary results show here are promising, but the ultimate performance and usability of backprojection interferometry is still unclear, especially in comparison to traditional methods. A comparison between results processed with the backprojection interferometry method and the traditional interferometry method could be completed, using backprojection processed images.

The UMass dual-frequency radar system is quite novel, given the limited existence of either S or Ka-band SAR systems, let alone a system with both frequencies. There is great potential to leverage the difference in scattering and penetration properties. Besides the current ongoing work developing and improving this system, which is not a minor task, there is likely room for a thorough theoretical look in to the potential scientific measurements which could be made.

BIBLIOGRAPHY

[1] Itzhack Y. Bar-Itzhack. "New Method for Extracting the Quaternion from a Rotation Matrix". In: *Journal of Guidance, Control, and Dynamics* 23 (2000), pp. 1085–1087.

[2] Roger R. Bate, Donald D. Mueller, and Jerry E. White. *Fundamentals of Astrodynamics*. New York, New York: Dover Publications, Inc., 1971.

[3] Amedeo Capozzoli, C. Curcio, and A. Liseno. "Fast GPU-Based Interpolation for SAR Backprojection". In: *Progress In Electromagnetics Research* 133 (2013), pp. 259–283.

[4] Walter G. Carrara, Ron S. Goodman, and Ronald M. Majewski. *Spotlight Synthetic Aperture Radar: Signal Processing Algorithms*. Norwood, Massachusetts: Artech House, Inc., 1995.

[5] Gerard Ruìz Carregal. "Design, fabrication and measurements of an s-band radar in an airborne platform". Masters thesis. Universitat Politècnica de Catalunya, 2014.

[6] *CUDA C Programming Guide*. `http://docs.nvidia.com/cuda/pdf/CUDA_C_Programming_Guide.pdf`. Accessed: 2016-06-14. Sept. 2015.

[7] Ian G. Cumming and Frank H. Wong. *Digital Processing of Synthetic Aperture Radar Data: Algorithms and Implementation*. Norwood, Massachusetts: Artech House, Inc., 2005.

[8] Michael I. Duersch and David G. Long. "Backprojection SAR interferometry". In: *International Journal of Remote Sensing* 36.4 (2015), pp. 979–999.

[9] Karsten Fourmont. "Non-Equispaced Fast Fourier Transforms with Applications to Tomography". In: *The Journal of Fourier Analysis and Applications* 9.5 (2003), pp. 431–450.

[10] Kan Fu, Paul Siqueira, and Rockwell Schrock. "A university-developed 35 GHz airborne cross-track SAR interferometer: Motion compensation and ambiguity reduction". In: *2014 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*. 2014.

[11] Herbert Goldstein. *Classical Mechanics*. Reading, Massachusetts: Addison-Wesley Publishing Company, Inc., 1980.

[12] Sir William Rowan Hamilton. "On quaternions; or on a New System of Imaginaries in Algebra". In: *The London, Edinburgh and Dublin Philosophical Magazine and Journal of Science (3rd Series)* 25 (1844).

[13] Scott Hensley et al. "Improved Processing of AIRSAR Data Based on the GeoSAR Processor". In: *2002 AIRSAR Earth Science and Application Workshop*. 2002.

[14] *NVIDIA's Next Generation CUDA Compute Architecture: Kepler GK110/210*. `http://images.nvidia.com/content/pdf/tesla/NVIDIA-Kepler-GK110-GK210-Architecture-Whitepaper.pdf`. Accessed: 2016-06-14. 2014.

[15] Pau Prats et al. "Comparison of Topography- and Aperture-Dependent Motion Compensation Algorithms for Airborne SAR". In: *IEEE Geoscience and Remote Sensing Letters* 4.3 (July 2007), pp. 349–353.

[16] Angel Ribalta. "Time-Domain Reconstruction Algorithms for FMCW-SAR". In: *IEEE Geoscience and Remote Sensing Letters* 8.3 (May 2011), pp. 396–400.

[17] Paul A. Rosen et al. "Synthetic Aperture Radar Interferometry". In: *Proceedings of the IEEE* 88.3 (Mar. 2000), pp. 333–382.

[18] Paul A. Rosen et al. "UAVSAR: A New NASA Airborne SAR System for Science and Technology Research". In: *2006 IEEE Conference on Radar*. 2006.

[19] Mehrdad Soumekh. *Synthetic Aperture Radar Signal Processing with MATLAB Algorithms*. New York, New York: John Wiley & Sons, Inc., 1999.

[20] Craig Stringham and David G. Long. "GPU Processing for UAS-Based LFM-CW Stripmap SAR". In: *Photogrammetric Engineering & Remote Sensing* 80.12 (Dec. 2014), pp. 1107–1115.

[21] Wolfgang Torge. *Geodesy*. Berlin: De Gruyter, 2001.

[22] Ridha Touzi et al. "Coherence Estimation for SAR Imagery". In: *IEEE Transactions on Geoscience and Remote Sensing* 37.1 (Jan. 1999), pp. 135–149.

[23] Hugues Vermeille. "Direct transformation from geocentric coordinates to geodetic coordinates". In: *Journal of Geodesy* 76 (Nov. 2002), pp. 451–454.