	2006415 GB-SAR 시스템 영상화 기법 개발 강원대학교	표지

주 의
(편집순서 8)

**(15 포인트
고딕계열)**

↑
6cm
↓

2006415

GB-SAR 시스템 영상화 기법 개발

Development of Imaging Technique for the GB-SAR System

강 원 대 학 교

한국지질자원연구원

제 출 문

한국지질자원연구원 원장 귀하

이 보고서를 “GB-SAR 시스템 영상화 기법 개발” 과제의 보고서로 제출합니다.

2006. 12

위탁연구기관명 : 강원대학교

위탁연구책임자 : 이 훈 열

연 구 원 : 채 희 삼

한 향 선

보고서 초록

<table>
<tr><td>과제고유번호</td><td>2006415</td><td>당해연도
연구기간</td><td>2006. 5. 1 -
2006. 12. 31</td><td>연차 구분</td><td>1년 차 / 총 1년</td></tr>
<tr><td>연구사업명</td><td colspan="5">기본연구사업</td></tr>
<tr><td rowspan="2">연구과제명</td><td>주관과제명</td><td colspan="4">지하 정밀 영상화 융합기술 개발</td></tr>
<tr><td>위탁과제명</td><td colspan="4">GB-SAR 시스템 영상화 기법 개발</td></tr>
<tr><td>연구책임자</td><td>이훈열</td><td>참여연구원수</td><td>총 : 3 명
내부 : 1 명
외부 : 2 명</td><td>연구비</td><td>10,000천원</td></tr>
<tr><td>연구기관명
및
소속부서명</td><td colspan="2">강원대학교 지구물리학과</td><td>참여기업명</td><td colspan="2"></td></tr>
<tr><td>국제공동연구</td><td colspan="5">상대국명 : 상대국연구기관명 :</td></tr>
<tr><td colspan="4">요약(연구결과를 중심으로 개조식 500자 이내)</td><td>보고서면수</td><td>64</td></tr>
<tr><td colspan="6">○ GB-SAR (Ground-Based Synthetic Aperture Radar) 시스템 설계
○ Range-Doppler 및 Deramp-FFT SAR 영상화 알고리즘 구현
○ 실내 및 실외 자료 시범 획득
○ GB-SAR 시스템의 이론적 최대해상도 구현
○ 간섭영상을 통한 InSAR, $\triangle k$-InSAR, DInSAR 가능성 확인
○ 시스템 보강 후 지반침하 및 사면 안정성 정밀 장기 측정 실용화 가능</td></tr>
<tr><td rowspan="2">색 인 어
(각 5개
이상)</td><td>한 글</td><td colspan="4">GB-SAR, 영상화 알고리즘, 간섭영상</td></tr>
<tr><td>영 어</td><td colspan="4">GB-SAR, imaging algorithm, interferogram</td></tr>
</table>

목 차

제 1 장 위탁연구과제의 개요
1.1 연구개발의 목적
1.2 연구개발의 필요성
1.3 연구개발의 범위
제 2 장 국내외 기술개발 현황
제 3 장 위탁연구사업의 내용 및 결과
3.1 GB-SAR 시스템 구성
3.2 GB-SAR 영상화 기법
3.3 GB-SAR 간섭 기법
3.4 GB-SAR 현장 실험
3.5 결론
제 4 장 목표달성도 및 관련분야에의 기여도
제 5 장 연구개발결과의 활용계획
제 6 장 위탁연구과정에서 수집한 해외과학기술정보
제 7 장 참고문헌
부록 A. GB-SAR 프로그램 원시코드

제 1 장 위탁연구과제의 개요

1.1 연구개발의 목적

이 연구의 목적은 GB-SAR 시스템을 설계하고 제작하여 자료를 시험 획득하고, GB-SAR 영상화 기법(focusing)과 간섭기법(interferometry)을 구현하는 것을 목표로 한다.

1.2 연구개발의 필요성

GB-SAR 시스템은 위상의 변화를 장기간 관측하여 지반 침하량 계측, 사면 안정성 평가, 눈사태 감지와 같은 인공 및 자연 구조물의 안정성을 판단하는 목적으로 사용될 수 있으며, 그 기술이 개발 단계에 있다. 이 연구에서 개발하고자 하는 GB-SAR는 항공기 및 인공위성에서 탑재되어 운용되는 SAR 시스템의 원리를 이용하여 지상에서 산란체의 마이크로파 반사 신호를 장기간 안정적으로 측정할 수 있는 시스템으로서, 국내에서는 최초로 시도되는 것이며 초기 기술 투자로 경쟁력을 확보할 수 있는 유리한 시점에 있다. GB-SAR는 항공기 및 인공위성 SAR의 지상 검보정 및 개념 설계용으로 사용될 수 있을 뿐만이 아니라, 독자적으로도 새로운 응용성을 찾을 수 있는 유망한 기술이다.

1.3 연구개발의 범위

○ GB-SAR (Ground-Based Synthetic Aperture Radar) 시스템 설계
○ Range-Doppler 및 Deramp-FFT SAR 영상화 알고리즘 구현
○ 실내 및 실외 자료 시범 획득
○ 간섭영상을 통한 InSAR, △k-InSAR, DInSAR, PSInSAR 가능성 확인

제 2 장 국내외 기술개발 현황

○ GB-SAR는 영국(SCEOS, 2005), 이탈리아(Noferini et. al., 2005), 일본(Hamasaki et. al., 2004) 등을 비롯한 전 세계 몇몇 연구실에서 갖추어 운영 중에 있다.

○ 현재까지 국내에서는 GB-SAR 시스템의 제작이 시도된 바 없다.

○ 2006년 1월 발사된 ALOS PALSAR, 2007년 2월에 발사 예정된 TerraSAR, 2007년 3월 발사 예정인 Radarsat-2, 그리고 2009년 우리나라의 Kompsat-5 인공위성에 탑재되는 SAR를 비롯한 차세대 SAR 시스템에서 얻어지는 고해상도, 다편광 영상을 통해 SAR 활용 기술이 지속해서 발전해 나아갈 것이다.

제 3 장 위탁연구사업의 내용 및 결과

3.1. GB-SAR 시스템 구성

GB-SAR 시스템은 크게 RF 부분과 Motion 부분으로 나눌 수 있다. RF 부분은 주로 네트워크 분석기, 마이크로파 안테나, 노트북 컴퓨터로 이루어져 있고, 이들을 연결하는 GPIB-USB, 마이크로파 케이블과 커넥터, Calibration Kit가 있다. Motion 부분은 레일 시스템, 모션 컨트롤러, Step motor, 랙 & 피니언 기어, 레일 연결부, 수평유지 나사 등으로 이루어져 있다(그림 1).

안테나는 실험의 목적에 맞도록 MTG Inc.(www.mtginc.co.kr)에서 설계 및 제작 후 특성 시험을 거쳐 납품되었다. Dual-Mode Transducer를 이용한 Dual-Polarization Square Horn Antenna로서, 포트가 2개 있어 동시에 Quadratic Polarization을 동시에 얻을 수 있다. 안테나는 5.0-5.6GHz 대역으로서, 중심 주파수가 5.3GHz로서 C-band인 Radarsat-1에 부합되도록 설계되었다. 구체적인 안테나 특성은 Table 1과 같다.

실험에 사용된 네트워크분석기(Network Analyzer)는 Agilent 8753ES로서 6GHz까지 사용이 가능하다. Calibration kit을 이용하여 시스템을 보정할 수 있으며, time-domain으로의 변환이 가능하다.

네트워크 분석기와 노트북을 National Instrument사의 GPIB-USB 커넥터로 연결하고 노트북에 설치된 LabView 프로그램을 통해 측정간격, 측정시간 그리고 자료저장을 제어하였다.

Motion 부분은 2m 길이의 레일을 3개 연결하여 총 6m의 레일을 만들었다. 모션 컨트롤러 및 스텝 모터를 이용하여 안테나의 수평적, 수직적 이동이 가능하다.

Table 1. 안테나 특성.

품 명	항목		규격	시험결과
Dual-Polarimetric Square Horn Antenna	Frequency(GHz)		5.0~5.6	5.0~5.6
	Beam Width	E-Plane(deg)	15(nominal)	12.1~13
		H-Plane(deg)	15(nominal)	15.5~17.5
	Gain(dBi)		20(nominal)	20.5 min.(20.5~21.8)
	VSWR		1.5 max.	1.4 max.
	Isolation		35dB 이하	50dB 이하
	Weight(kg)		4 max.	3.1 max.
	Input Impedance(Ω)		50	50

그림 1. GB-SAR 시스템 구성

3.2. GB-SAR 영상화 기법

GB-SAR 시스템은 안테나가 레일 위를 이동하면서 합성 구경(synthetic aperture)을 이루어 고해상도의 SAR 영상화(SAR Focusing)를 이룬다. 보통의 항공기 및 인공위성 SAR는 탑재체가 빠른 속도로 움직이면서 펄스를 연속적으로 발사하게 된다. 이 때, 펄스가 안테나에서 송신되어 산란체를 맞고 반사되어 다시 안테나로 수신될 때까지의 시간(fast time) 동안에도 탑재체의 움직임(slow time)이 있기 때문에 신호 처리가 다소 복잡해질 수 있다. 마이크로파의 전파 속도에 비하여 탑재체의 움직임이 매우 느리다는 것을 이용하여 두 시간을 서로 분리해 신호처리를 한다. 그러나 GB-SAR에서는 스텝 모터를 이용하여 레일 위에 안테나가 정지해 있는 상태에서 송수신이 이루어지기 때문에, 이러한 가정이 필요가 없게 되어 보다 정밀한 영상화가 가능해진다.

GB-SAR 시스템은 합성 구경의 길이가 레일의 길이로 제한되어 있다는 점에서 또한 항공기 및 인공위성 SAR와는 크게 다르다. 항공기 및 위성 SAR에서는 합성 구경의 길이가 안테나의 빔 폭과 같다는 의미에서 full focusing이라 할 수 있으며, 해상도는 거리에 상관없이 안테나 길이의 절반으로 최적화 된다. 그러나 GB-SAR는 레일에 근접한 부분을 제외한 대부분의 이미지 영역에서 합성 구경이 레일의 길이로 제한된다는 의미에서 partial focusing이라 할 수 있으며, 해상도 역시 대부분 영역에서 거리의 함수가 된다. 또한 자료처리의 효율성을 위하여 보통의 Range-Doppler Algorithm외에도 Deramp-FFT Algorithm과 같은 SAR

Focusing Algorithm을 검토할 필요가 있다.

여기서는 GB-SAR 신호를 먼저 Range compression 한 다음 Azimuth Compression하는 과정에서 Deramp-FFT 및 Range-Doppler Algorithm을 모두 구현하여 비교 검토해 보았다.

(1) Range Compression

네트워크 분석기는 frequency sweep을 통하여 자료를 얻은 후 이를 time-domain으로 역푸리에 변환(inverse Fourier transformation)하여, 마치 시간적으로 짧은 sinc 소스를 발사하여 되돌아오는 신호를 얻는 레이더의 효과를 얻는다. 이를 수식으로 정리하면 다음과 같다.

네트워크 분석기는 주파수 영역에서 작동하기 때문에, 위 수식을 푸리에 변환(Fourier transformation)을 통해 나타나는 아래 수식과 같은 원리로 작동한다.

$$S(f) = H(f)G(f) \tag{1}$$

이 때, $H(f)$는 중심이 f_c 이고 밴드 폭이 B_R 인 box function이다. 즉, 안테나는 네트워크 분석기가 만들어낸 특정 진동수인 f를 continuous wave 형태로 쏘아 보내고, 이 단일 진동수의 마이크로파가 목표물 전체에 반응하는 후방산란 계수 $G(f)$를, 다시 안테나에 돌아오는 신호를 통해 얻게 된다. 네트워크 분석기는 계속하여 진동수를 증가시키면서(frequency sweep) 주어진 진동수 영역에서 $G(f)$를 모두 얻게 된다. 이 때, calibration이 잘 된 네트워크 분석기의 $H(f)$ 값은 주어진 밴드 폭에서 1의 값을 가지게 되므로 frequency sweep을 통해 식(1)의 $S(f)$ 함수가 얻어진다. 이를 역푸리에 변환을 통해 시간 영역으로 변환시키면 다음과 같은 레이더의 작동 원리가 구현된다.

$$s(t) = h(t) * g(t) \tag{2}$$

만약 목표물이 $t = t_0 = 2R/c$에 있으며 $g(t) = \delta(t - t_0)$, 즉 한 점(point target)이라면 레이더에서 얻어진 값은 식 (2)에 의하여 다음과 같이 된다.

$$s(t) = B_R \mathrm{sinc}[\pi B_R (t - t_0)] \exp[j 2\pi f_c (t - t_0)] \tag{3}$$

따라서 네트워크 분석기에서 시간 영역으로 변환된 측정값의 크기와 위상 ϕ는 다음과 같다.

$$|s(t)| = |\mathrm{sinc}[\pi B_R (t - t_0)]| = |\mathrm{sinc}[\pi B_R (t - 2R/c)]| \tag{4}$$

$$\phi = 2\pi f_c (t - t_0) = 2\pi f_c (t - 2R/c) \tag{5}$$

즉, 신호의 크기는 $t = t_0 = 2R/c$에서 최대값을 나타내며, 위상은 안테나와 목표물간의 거리의 함수가 된다. 식(5)에서 시스템에서 위상 측정 오차를 ϕ_{error}라고 한다면, 거리 측정의 오차 R_{error}는 다음과 같다.

$$R_{error} = \frac{\lambda}{4\pi}\phi_{error} \tag{6}$$

시스템과 목표물이 안정된 상태에서 수차례 위상을 측정 했을 때 그 변화가 1도 정도라면 $\phi_{error} = 1^\circ$가 되고, 파장 $\lambda = 5.6cm$인 시스템에서 $R_{error} = 0.08mm$ 정도가 되어 매우 정밀한 거리(slant range) 값을 줄 수 있으며, 이것이 DInSAR의 정밀도를 주게 된다. 보통 위성 시스템에서는 위성체의 잡음이나 측정 시간에 따른 차이로 인하여 최악의 경우 $\phi_{error} = 10^\circ$로 보고 있으며, 그렇다 하더라도 $R_{error} = 0.8mm$가 되어 매우 정밀한 지표변위를 측정할 수 있는 이론적 근거가 마련된다.

식 (3)에 있는 sinc 함수의 크기가 최대값에서 3dB 떨어지는 시간을 보통 레이더의 시간 해상도 δt로 정의하며, 이는 다음과 같다.

$$\delta t = 1/B_R \tag{7}$$

예를 들어 $f_c = 5.3GHz$ 이고, $B_R = 600MHz$이면 시간 해상도는 $\delta t = 1.67ns$이다. 이를 안테나와 목표물과의 왕복거리를 고려하면 거리 해상도(slant range resolution) δR은 다음과 같다.

$$\delta R = c\delta t/2 = c/(2B_R) \tag{8}$$

따라서 위의 예에서는 $\delta R = 25cm$ 에 해당한다. 즉, slant range 방향으로 25cm 떨어진 두 물체를 구별할 수 있는 해상도를 가진다고 할 수 있다. 이 레이더 시스템의 최대 거리 $R_{\max}$는 네트워크 분석기의 frequency sampling 수가 N_R일 때, 다음과 같이 정의된다.

$$R_{\max} = (N_R - 1)\delta R \tag{9}$$

Agilent 8753ES 네트워크 분석기의 최대 N은 1601이며, 이 경우 $R_{\max} = 400m$이다. 이러한 최대 거리는 frequency sweep 밴드 폭을 줄이면 식(7)-(9)에 의해 무한대까지 늘어날 수 있다. 그러나 네트워크 분석기의 파워가 10dBm으로 제한되어 있기 때문에 최대 거리에는 한계가 있다. 이러한 한계를 극복하기 위하여 신호증폭기를 추가하여 설치할 수 있다. 발신기와 수신기가 하나의 안테나로 이루어지는 이 시스템에서는 신호증폭기의 설치가 불가하기 때문에 송신과 수신 안테나를 따로 두어야 한다.

여기서 주의할 점은, 네트워크 분석기를 이용한 frequency sweep을 하는 데 보통 수 초에서 수십 초가 걸리므로, 그 동안 시스템을 구성하는 안테나, 지지대, 그리고 목표물의 움직임이 심한 경우에는 레이더 작동 원리를 구현할 수 없는 단점이 있다.

(2) Azimuth Compression

SAR 시스템에서 Azimuth 해상도를 구현하는 방법에는 Range-Doppler Algorithm, Chirp Scaling Algorithm, Omega-K Algorithm, Deramp-FFT (또는 SPECAN) Algorithm 등 여러 가지가 있다. 보통 항공기나 인공위성과 같이 탑재체의 연속적인 움직임을 이용하는 strip mode에서는 Range-Doppler Algorithm이 보편적으로 사용된다. 그러나 scan mode나 burst mode와 같이 azimuth 방향으로 펄스가 연속적이지 않고 군집을 이루는 경우에는 Deramp-FFT Algorithm을 사용하는 것이 처리 시간과 메모리를 절약할 수 있다.

GB-SAR 시스템은 레일의 길이에 따라 움직임이 제한되어 있다는 점에서 여타 SAR 시스템과 다르다. GB-SAR 시스템에서 효율적인 Focusing Algorithm은 거리(range)에 따라 결정된다. 레일의 길이가 고정된 상태에서, Azimuth 방향으로의 빔폭이 레일보다 작은 근거리에서는 Range-Doppler Algorithm이 정확한 Focusing 결과를 내 주지만, 원거리에서는 Deramp-FFT Algorithm이 효율적이다. 따라서 이 연구에서는 GB-SAR 시스템의 용도가 근거리, 혹은 원거리일 경우를 대비하여 두 알고리즘을 모두 프로그램으로 구현하였다.

GB-SAR는 레일을 통하여 Azimuth 방향으로 안테나를 움직이면서 자료를 얻고 신호처리를 통하여 Azimuth 방향으로 높은 해상도를 구현한다는 점에서 여타 SAR 시스템과 비슷하다. 그러나 탑재체가 정지한 상태에서 자료를 얻을 수 있기 때문에 안테나를 통해 나가는 마이크로파에 도플러 효과가 나타나지 않기 때문에 zero-Doppler 영상을 얻을 수 있는 장점이 있다.

Deramp-FFT Algorithm

SAR Focusing 기법 중에서 주로 처리 시간과 메모리가 적게 드는 deramp-FFT 기법을 먼저 알아보자. 한 점 (R_0, x_c)에 위치한 산란체에서 얻어진 자료에 대한 처리는 다음과 같다. GB-SAR 원시 자료를 base-banded 후 iFFT를 하여 range compression 을 실시하면 다음과 같은 형태가 된다.

$$s(x|R_0, x_c) = e^{-j\frac{4\pi}{\lambda}R(x|R_0, x_c)}, \ |x - x_c| < R_0\frac{\lambda}{L_x} \text{ and } |x| \le X/2 \tag{10}$$

그림 2에 의하면 빔의 폭과 레일의 길이, 그리고 목표점의 위치에 따라 x의 정의 구역, 즉 산란체가 펄스를 받는 레일의 위치가 달라진다. 안테나와 산란체의 거리의 함수는 다음과 같다.

$$R = \sqrt{R_0^2 + (x - x_c)^2}\,. \tag{11}$$

이를 $x = 0$ (레일의 중심)을 중심으로 Taylor Series를 적용하면,

$$R(x|R_0, x_c) = R(0) + R'(0)x + \frac{R''(0)}{2!}x^2 + \cdots \tag{12}$$

와 같이 근사할 수 있다. 이를 계산하면,

$$R(0) = \sqrt{R_0^2 + x_c^2} = R_c, \tag{13}$$

$$R'(x) = \frac{x - x_c}{R(x)}, \; R'(0) = -\frac{x_c}{R(0)} = -\frac{x_c}{R_c}, \tag{14}$$

$$R''(x) = \frac{R(x) - (x - x_c)R'(x)}{R^2(x)}, \tag{15}$$

$$R''(0) = \frac{R(0) + x_c R'(0)}{R^2(0)} = \frac{R_c + x_c(-x_c/R_c)}{R_c^2} = \frac{R_c^2 - x_c^2}{R_c^3} = \frac{R_0^2}{R_c^3} \tag{16}$$

와 같이 된다. 따라서,

$$R = R_c - \frac{x_c}{R_c}x + \frac{R_0^2}{2R_c^3}x^2 + \cdots \tag{17}$$

이다. 그러므로

$$s(x|R_0, x_c) = e^{-j\frac{4\pi}{\lambda}\left(R_c - \frac{x_c}{R_c}x + \frac{R_0^2}{2R_c^3}x^2 + \cdots\right)} \tag{18}$$

로 나타낼 수 있다.

Deramp-FFT는 먼저 단일 목표점에 대한 거리의 함수를 이용하여 Range 방향으로 정렬하며, 이러한 과정을 Range Migration이라고 한다. 그 후 다음의 deramp function을 수신 신호에 곱하고 이를 FFT하는 순서로 이루어진다.

$$h^{-1}(x) = e^{j\frac{4\pi}{\lambda}\left(\frac{R_0^2}{2R_c^3}x^2\right)} \tag{19}$$

이를 수신 신호(10)에 곱하고 x에 대한 wavenumber 영역 u로 FFT를 수행하면 Azimuth Compression이 수행된다.

$$g(u) = \int_{-\infty}^{\infty} s(x)h^{-1}(x)e^{-j2\pi ux}dx \tag{20}$$

여기서 수신 신호의 정의구역은 $[C - L_s/2, C + L_s/2]$로 나타낼 수 있다. 여기서 C는 레일에서 산란체를 비추는 안테나의 중심 위치, L_s는 안테나의 해당 레일 구간의 폭(합성구경 길이)을 의미하며, 산란체가 위치한 영역에 따라 다르다(그림 2). 이를 적용하면,

$$g(u) = \int_{C - L_s/2}^{C + L_s/2} e^{-j\frac{4\pi}{\lambda}\left(R_c - \frac{x_c}{R_c}x\right)} e^{-j2\pi ux}dx$$

$$= e^{-j\frac{4\pi}{\lambda}R_c} e^{-j2\pi\left(u - \frac{2x_c}{\lambda R_c}\right)C} \int_{-L_s/2}^{L_s/2} e^{-j2\pi\left(u - \frac{2x_c}{\lambda R_c}\right)x'} dx'$$

$x' = x - C$로 치환하여 계산하면,

$$\therefore g(u) = e^{-j\frac{4\pi}{\lambda}R_c} e^{-j2\pi\left(u - \frac{2x_c}{\lambda R_c}\right)C} L_s \mathrm{sinc}\left[\pi L_s\left(u - \frac{2x_c}{\lambda R_c}\right)\right] \tag{21}$$

위의 sinc 함수는 $u=\frac{2}{\lambda R_c}x_c$에서 최대 peak를 가지며 half power width로 정의되는 해상도는 $\delta u=\frac{1}{L_s}$을 가진다. 따라서 이 신호의 최대 진폭을 가지는 $u-$space는 x_c-space로 바로 해석이 가능하며, 그 관계는 $x_c=\frac{\lambda R_c}{2}u$가 되며, Azimuth 해상도는 $\delta x_c=\frac{\lambda R_c}{2}\delta u=\frac{\lambda R_c}{2L_s}$와 같이 된다. 실제 신호 처리는 FFT를 이용하게 되며, 이때 사용된 azimuth sample을 $N_{xfft}(=2^n \geq N_x)$라 하면, 가상의 레일 간격은 $X_{fft}=(N_{xfft}-1)\triangle x$와 같이 된다. 그리고 영상의 픽셀 간격은 $\triangle x_{image}=\frac{\lambda R_c}{2}\triangle u=\frac{\lambda R_c}{2X_{fft}}$와 같이 거리의 함수로 주어지기 때문에 SAR Focusing 후 resampling을 통하여 기하 보정이 필요하다. Azimuth 영상의 폭 또한 $x_{width}=(N_{xfft}-1)\triangle x_{image}=\frac{\lambda R_c}{2\triangle x}$로서, 거리의 함수이다.

그림 2의 각 영역별로 SAR Focusing에 대한 구체적인 사항은 아래와 같다.

가) Region I: Partial focusing with full rail scan (그림 3)

안테나 구간: $[-X/2, X/2]$

유효 적분 구간: $C=0, L_s=X$

$$g(u)=e^{-j\frac{4\pi}{\lambda}R_c}X\mathrm{sinc}\left[\pi X\left(u-\frac{2x_c}{\lambda R_c}\right)\right]$$

해상도: $\delta x=\frac{\lambda R_c}{2X}$

즉, 해상도는 R_c가 증가함에 따라 해상도가 나빠진다.

나) Region II: Partial focusing with partial rail scan (그림 4)

안테나 구간: $\left[x_c-\frac{\lambda R_0}{2L_x}, \frac{X}{2}\right]$

유효 적분 구간:

$$C=\frac{1}{2}\left(x_c-\frac{\lambda R_0}{2L_x}+\frac{X}{2}\right), \quad L_s=\frac{X}{2}-\left(x_c-\frac{\lambda R_0}{2L_x}\right)$$

$$g(u)=e^{-j\frac{4\pi}{\lambda}R_c}e^{-j2\pi\left(u-\frac{2x_c}{\lambda R_c}\right)\frac{1}{2}\left(x_c-\frac{\lambda R_0}{2L_x}+\frac{X}{2}\right)} \times\left[\frac{X}{2}-\left(x_c-\frac{\lambda R_0}{2L_x}\right)\right]\mathrm{sinc}\left\{\pi\left[\frac{X}{2}-\left(x_c-\frac{\lambda R_0}{2L_x}\right)\right]\left(u-\frac{2x_c}{\lambda R_c}\right)\right\}$$

해상도: $\delta x_c = \dfrac{\lambda R_c}{2\left[\dfrac{X}{2} - \left(x_c - \dfrac{\lambda R_0}{2L_x}\right)\right]}$

R_c가 증가함에 따라, 그리고 x_c가 커짐에 따라 해상도가 나빠진다.

다) Region III: Partial focusing with partial rail scan (그림 5)

안테나 구간: $\left[-\dfrac{X}{2}, x_c + \dfrac{\lambda R_0}{2L_x}\right]$

유효 적분 구간:

$$C = \frac{1}{2}\left(-\frac{X}{2} + x_c + \frac{\lambda R_0}{2L_x}\right),\ L_s = \left(x_c + \frac{\lambda R_0}{2L_x}\right) + \frac{X}{2}$$

$$g(u) = e^{-j\frac{4\pi}{\lambda}R_c} e^{-j2\pi\left(u - \frac{2x_c}{\lambda R_c}\right)\frac{1}{2}\left(-\frac{X}{2} + x_c + \frac{\lambda R_0}{2L_x}\right)} \times \left[x_c + \frac{\lambda R_0}{2L_x} + \frac{X}{2}\right] \mathrm{sinc}\left\{\pi\left[x_c + \frac{\lambda R_0}{2L_x} + \frac{X}{2}\right]\left(u - \frac{2x_c}{\lambda R_c}\right)\right\}$$

해상도: $\delta x_c = \dfrac{\lambda R_c}{2\left[\left(x_c + \dfrac{\lambda R_0}{2L_x}\right) + \dfrac{X}{2}\right]}$

R_0가 증가함에 따라, 그리고 x_c가 작아짐에 따라 나빠진다. 이는 Region II와 대칭을 이룬다.

라) Region IV: Full focusing with partial rail scan (그림 6)

안테나 구간: $\left[x_c - \dfrac{\lambda R_0}{2L_x}, x_c + \dfrac{\lambda R_0}{2L_x}\right]$

유효 적분 구간:

$$C = x_c,\ L_s = \frac{\lambda R_0}{L_x}$$

$$g(u) = e^{-j\frac{4\pi}{\lambda}R_c} e^{-j2\pi\left(u - \frac{2x_c}{\lambda R_c}\right)x_c} \frac{\lambda R_0}{L_x} \mathrm{sinc}\left\{\pi \frac{\lambda R_0}{L_x}\left(u - \frac{2x_c}{\lambda R_c}\right)\right\}$$

해상도: $\delta x = \dfrac{L_x}{2}\dfrac{R_c}{R_0} = \dfrac{L_x}{2}\sqrt{1 + (x_c/R_0)^2}$

이 경우에는 레일의 중심에서 수직인 방향에 있는 산란체($x_c = 0$)는 안테나 구경의 절반의 해상도를 가지게 된다. 그리고 여기에서 벗어나는 경우, 즉, x_c가 증가하면 해상도가 줄어든다. 이는 SAR strip mode의 전형적인 예에 해당하며 안테나 빔 폭이 허용하는 길이만큼 충분히 azimuth 방향으로 샘플링이 이루어지는 경우이다. 그러한 의미에서 이 영역의 SAR 영상의 특성을 Full Focusing이라 지칭할 수 있다.

Range Limit of Deramp-FFT Algorithm

위의 deramp 함수는 $R_c = R_c(R_0, x_c)$의 함수이므로 정확한 SAR focusing을 위해서는 R_0는 물론, x_c에 대해서도 서로 다른 deramp function과 FFT를 반복 수행하여야하며, 이는 시간과 비용이 time-domain azimuth compression과 동일하게 낭비된다. 따라서 deramp-FFT Algorithm은 위의 deramp function에서 $R_c \simeq R_0$라는 근사값을 사용하여 식 (19)를 대신한다.

$$\tilde{h}(x) = e^{j\frac{4\pi}{\lambda}\left(\frac{1}{2R_0}x^2\right)} = e^{j\frac{2\pi}{\lambda R_0}x^2} \tag{22}$$

을 사용한다. 이 함수는 x_c와는 무관하기 때문에, 단 한번의 deramp-FFT 과정으로도 SAR Focusing이 근사적으로 이루어져 시간비용이 대폭 줄어든다. 하지만, 이 때문에 아래와 같은 영상화에 제한이 따르게 된다.

Deramp-FFT algorithm에서는 $R_c \simeq R_0$근사 때문에 다음과 같은 phase error가 발생한다.

$$|\delta\phi| = \frac{4\pi}{\lambda}\left|\frac{R_0^2}{2R_c^3} - \frac{1}{2R_0}\right|x^2 = \frac{2\pi}{\lambda R_0}\left|\left(\frac{R_0}{\sqrt{R_0^2 + x_c^2}}\right)^3 - 1\right|x^2 \tag{23}$$

이 때, phase error의 최대값은 $x = \pm X/2$일때, 그리고 $x_c = \pm\frac{\lambda R_0}{2L_x}$(half beam footprint)에서 발생하며, Defocusing이 일어나지 않기 위해서는 phase error가 $\delta\phi_{crit}$보다 작아야 한다.

$$\begin{aligned} |\delta\phi|_{\max} &= \frac{2\pi}{\lambda R_0}\left|\left(\frac{R_0}{\sqrt{R_0^2 + \left(\frac{\lambda R_0}{2L_x}\right)^2}}\right)^3 - 1\right|\left(\frac{X}{2}\right)^2 \\ &= \frac{\pi}{2\lambda R_0}\left|\left(\frac{1}{\sqrt{1 + \left(\frac{\lambda}{2L_x}\right)^2}}\right)^3 - 1\right|X^2 \le \delta\phi_{crit} \end{aligned} \tag{24}$$

따라서 Defocusing이 발생하기 않기 위해서는 다음과 같은 거리 제한이 필요하다.

$$R_0 \ge \frac{\pi X^2}{2\lambda}\left|\left(\frac{1}{\sqrt{1 + \left(\frac{\lambda}{2L_x}\right)^2}}\right)^3 - 1\right|\frac{1}{\delta\phi_{crit}} \tag{25}$$

예를 들면 $f_c = 5.3\text{GHz}$, $\lambda = 0.056564623m$, $X = 5m$, $L_x = 0.25m$이면

$$R_0 \ge 13.12 \times \frac{1}{\delta\phi_{crit}} \tag{26}$$

이다. 예를 들어 $\delta\phi = \frac{\pi}{4}$로 잡으면(Curlander 1991, p.173), $R_0 \ge 16.71m$가 되어야 한다. 이보다 더 까다로운 조건인 $\delta\phi = 10^\circ = \frac{\pi}{18}$로 잡으면, $R_0 \ge 75.16m$가 된다. 따

라서 근거리에서는 deramp-FFT Algorithm으로는 영상화가 이루어지지 않으며, 이 경우에는 Range-Doppler Algorithm을 사용할 것이 권장된다.

Range-Doppler Algorithm

한 점 (R_0, x_c)에 위치한 산란체에 대한 Range-Doppler Algorithm 자료처리는 다음과 같다. 위에서와 마찬가지로 range compression 후 자료는 다음과 같다.

$$s(x|R_0, x_c) = e^{-j\frac{4\pi}{\lambda}R(x|R_0, x_c)}, \ |x - x_c| < R_0\frac{\lambda}{L_x} \ \text{and} \ |x| \le X/2 \tag{27}$$

또한 그림 2에 의하면,

$$R = \sqrt{R_0^2 + (x - x_c)^2} \tag{28}$$

이다. Deramp-FFT에서는 위 함수를 $x = 0$에서 근사한 반면, Range-Doppler Algorithm 에서는 $x = x_c$ 를 중심으로 Taylor Series를 적용한다.

$$R(x|R_0, x_c) = R(x_c) + R'(x_c)(x - x_c) + \frac{R''(x_c)}{2!}(x - x_c)^2 + \cdots \tag{29}$$

이 때 식 (27)은 다음과 같이 분석할 수 있다.

Phase: $\phi = -\frac{4\pi}{\lambda}R(x)$ (30)

Rate of phase (Wavenumber Doppler):

$$u_D \equiv \frac{\phi'}{2\pi} = -\frac{2}{\lambda}R'(x) \tag{31}$$

Wavenumber Doppler rate:

$$u_D{}'(x) = \frac{\phi''}{2\pi} = -\frac{2}{\lambda}R''(x) \tag{32}$$

식(29)를 wavenumber Doppler parameter로 바꾸어주면,

$$R(x|R_0, x_c) = R(x_c) - \frac{\lambda u_{Dc}}{2}(x - x_c) - \frac{\lambda u_R}{4}(x - x_c)^2 + \cdots \tag{33}$$

이다. 그림 2의 GB-SAR 시스템에서 이를 계산하면,

$$R(x_c) = R_0 \tag{34}$$

$$R'(x) = \frac{x - x_c}{R(x)}, \quad R'(x_c) = 0 \tag{35}$$

$$R''(x) = \frac{R(x) - (x - x_c)R'(x)}{R^2(x)}, \qquad R''(x_c) = \frac{R(x_c)}{R^2(x_c)} = \frac{1}{R_0} \tag{36}$$

이므로,

$$u_{Dc} = 0, \tag{37}$$

$$u_R = -\frac{2}{\lambda R_0} \tag{38}$$

가 된다. 따라서 식(27)은

$$s(x|R_c, x_c) = e^{-j\frac{4\pi}{\lambda}R(x_c)} e^{j2\pi[u_{Dc}(x-x_c)+u_R(x-x_c)^2/2]},\ |x-x_c| < R_0\frac{\lambda}{L_x} \text{ and } |x| \le X/2 \quad (39)$$

로 표현된다. 이 신호는 azimuth linear chirp을 형성한다. Range-Doppler Algorithm 에서는 영상화를 위한 filter로서 다음을 사용한다.

$$h^{-1}(x) = e^{-j2\pi[u_{Dc}x + u_R x^2/2]},\ |x| < \frac{\lambda R_0}{2L_x} \quad (40)$$

이를 이용하여 다음의 matched filtering을 수행하면 SAR focusing이 이루어진다.

$$g(x) = \int_{-\infty}^{\infty} h^{-1}(x'-x)s(x')dx' \quad (41)$$

해당 적분구간의 중심을 C, 그 폭을 합성구경의 길이인 L_s라고 할 때, 적분 구간은 $[C-L_s/2, C+L_s/2]$로 나타낼 수 있다. 여기서 C는 레일에서 산란체를 비추는 안테나의 중심 위치, L_s는 안테나의 해당 레일 구간의 폭을 의미한다(그림 2).

$$g(x) = \int_{C-L_s/2}^{C+L_s/2} e^{-j2\pi[u_{Dc}(x'-x)+u_R(x'-x)^2/2]} e^{-j\frac{4\pi}{\lambda}R(x_c)} e^{j2\pi[u_{Dc}(x'-x_c)+u_R(x'-x_c)^2/2]} dx'$$

$$= e^{-j\frac{4\pi}{\lambda}R(x_c)} e^{j2\pi[u_{Dc}(x-x_c)-u_R(x^2-x_c^2)/2]} \int_{C-L_s/2}^{C+L_s/2} e^{j2\pi u_R(x-x_c)x'} dx' \quad (42)$$

$x'' = x' - C$로 치환하면,

$$g(x) = e^{-j\frac{4\pi}{\lambda}R(x_c)} e^{j2\pi[u_{Dc}(x-x_c)-u_R(x^2-x_c^2)/2]} \int_{-L_s/2}^{L_s/2} e^{j2\pi u_R(x-x_c)(x''+C)} dx''$$

$$= e^{-j\frac{4\pi}{\lambda}R(x_c)} e^{j2\pi[u_{Dc}(x-x_c)-u_R(x^2-x_c^2)/2+u_R(x-x_c)C]} \int_{-L_s/2}^{L_s/2} e^{j2\pi u_R(x-x_c)x''} dx''$$

$$= e^{-j\frac{4\pi}{\lambda}R(x_c)} e^{j2\pi[u_{Dc}(x-x_c)-u_R(x^2-x_c^2)/2+u_R(x-x_c)C]} L_s \operatorname{sinc}\left[\pi L_s u_R(x-x_c)\right] \quad (43)$$

즉, GB-SAR Focusing후 신호의 크기는 $x = x_c$ 에서 최대값 L_s를 가지며 sinc 함수 최대값의 -3dB 를 기준으로 해상도를 정의하면 $\delta x = \frac{1}{L_s|u_R|} = \frac{1}{B_x}$ 이다. 여기서 빔의 중심부나 강한 후방산란을 보이는 산란체에서는 sinc 함수의 영향이 azimuth 방향으로 연속적으로 보일 수 있으며, 이 때 phase는 거리에 의한 함수뿐만이 아니라 위 식의 두 번째 phase term도 interferogram에 나타나게 된다.

여기서 GB-SAR 시스템인 $R(x_c) = R_0$ $u_{Dc} = 0$, $u_R = -\frac{2}{\lambda R_0}$을 적용하면,

$$\therefore g(x) = e^{-j\frac{4\pi}{\lambda}R_0} e^{j2\pi\left[\frac{2}{\lambda R_0}(x-x_c)\left(\frac{x+x_c}{2}-C\right)\right]} L_s \operatorname{sinc}\left[\frac{2\pi L_s}{\lambda R_0}(x-x_c)\right] \quad (44)$$

(sinc는 even function)가 된다. 마찬가지로 $x = x_c$에서 최대값 L_s를 가지며, azimuth bandwidth는 $B_x = \frac{2L_s}{\lambda R_0}$, 해상도는 $\delta x = \frac{\lambda R_0}{2L_s}$이다. phase는 $\phi = -\frac{4\pi}{\lambda} R_0$가 되는데, 이는 Deramp-FFT결과가 $\phi = -\frac{4\pi}{\lambda} R_c$인 것과 대별된다. 실제 신호 처리는 FFT를 이용하게 되며, 이 때 사용된 azimuth sample을 $N_{xfft} (= 2^n \geq N_x)$라 하면, 가상의 레일 간격은 $X_{fft} = (N_{xfft} - 1)\triangle x$와 같이 된다. 그리고 영상의 azimuth 픽셀 간격은 $\triangle x_{image} = \triangle x$와 같이 되며, azimuth 영상의 폭 또한 $x_{width} = (N_{xfft} - 1)\triangle x$로서 거리에 무관하여, Deramp-FFT와는 달리 기하 보정이 필요 없다.

그림 2의 영여을 고려 할 때, 형성되는 영상은 아래의 네 부분으로 나눌 수 있다.

가) Region I: Partial focusing with full rail scan (그림 3)

안테나 구간: $[-X/2, X/2]$

유효 적분 구간: $C = 0, L_s = X$

$$g(x) = Xe^{-j\frac{4\pi}{\lambda}R_0} e^{j\frac{2\pi}{\lambda R_0}(x^2 - x_c^2)} \text{sinc}[\frac{2\pi}{\lambda R_0} X(x - x_c)]$$

Azimuth Bandwidth: $B_a = \frac{2X}{\lambda R_0}$

해상도: $\delta x = \frac{\lambda R_0}{2X}$

해상도는 R_0가 증가함에 따라 나빠진다.

나) Region II: Partial focusing with partial rail scan (그림 4)

안테나 구간: $\left[x_c - \frac{\lambda R_0}{2L_x}, \frac{X}{2}\right]$

유효 적분 구간:

$$C = \frac{1}{2}\left(x_c - \frac{\lambda R_0}{2L_x} + \frac{X}{2}\right), \quad L_s = \frac{X}{2} - \left(x_c - \frac{\lambda R_0}{2L_x}\right)$$

$$g(x) = e^{-j\frac{4\pi}{\lambda}R_0} e^{j2\pi\left\{\frac{2}{\lambda R_0}(x - x_c)\left[\frac{x + x_c}{2} - \frac{1}{2}\left(x_c - \frac{\lambda R_0}{2L_x} + \frac{X}{2}\right)\right]\right\}}$$
$$\times \left[\frac{X}{2} - \left(x_c - \frac{\lambda R_0}{2L_x}\right)\right] \text{sinc}\left\{\frac{2\pi}{\lambda R_0}\left[\frac{X}{2} - \left(x_c - \frac{\lambda R_0}{2L_x}\right)\right](x - x_c)\right\}$$

Azimuth Bandwidth: $B_a = \frac{2}{\lambda R_0}\left(\frac{X}{2} - x_c\right) + \frac{1}{L_x}$,

해상도: $\delta x = \frac{1}{B_a} = \frac{1}{\frac{2}{\lambda R_0}\left(\frac{X}{2} - x_c\right) + \frac{1}{L_x}}$

즉, R_0가 증가함에 따라, 그리고 x_c가 커짐에 따라 나빠진다. $\lambda = 0.0566m$, $X = 5m$일 경우 거리에 따른 해상도는 다음과 같다.

다) Region III: Partial focusing with partial rail scan (그림 5)

안테나 구간: $\left[-\frac{X}{2}, x_c + \frac{\lambda R_0}{2L_x}\right]$

유효 적분 구간:

$$C = \frac{1}{2}\left(-\frac{X}{2} + x_c + \frac{\lambda R_0}{2L_x}\right),\ L_s = \left(x_c + \frac{\lambda R_0}{2L_x}\right) + \frac{X}{2}$$

$$g(x) = e^{-j\frac{4\pi}{\lambda}R_0} e^{j2\pi\left\{\frac{2}{\lambda R_0}(x - x_c)\left[\frac{x + x_c}{2} - \frac{1}{2}\left(-\frac{X}{2} + x_c + \frac{\lambda R_0}{2L_x}\right)\right]\right\}} \times \left[\left(x_c + \frac{\lambda R_0}{2L_x}\right) + \frac{X}{2}\right] \mathrm{sinc}\left\{\frac{2\pi}{\lambda R_0}\left[\left(x_c + \frac{\lambda R_0}{2L_x}\right) + \frac{X}{2}\right](x - x_c)\right\}$$

Azimuth Bandwidth: $B_a = \frac{2}{\lambda R_0}\left(\frac{X}{2} + x_c\right) + \frac{1}{L_x}$,

해상도: $\delta x = \frac{1}{B_a} = \frac{1}{\frac{2}{\lambda R_0}\left(\frac{X}{2} + x_c\right) + \frac{1}{L_x}}$

즉, R_0가 증가함에 따라, 그리고 x_c가 작아짐에 따라 나빠진다. 이는 Region II와 대칭을 이루는 구조이다.

라) Region IV: Full focusing with partial rail scan (그림 6)

안테나 구간: $\left[x_c - \frac{\lambda R_0}{2L_x}, x_c + \frac{\lambda R_0}{2L_x}\right]$

유효 적분 구간:

$$C = x_c,\ L_s = \frac{\lambda R_0}{L_x}$$

$$g(x) = e^{-j\frac{4\pi}{\lambda}R_0} e^{j2\pi\left[\frac{2}{\lambda R_0}(x - x_c)\left(\frac{x + x_c}{2} - x_c\right)\right]} \frac{\lambda R_0}{L_x} \mathrm{sinc}\left[\frac{2}{L_x}(x - x_c)\right]$$

Azimuth Bandwidth: $B_a = \frac{2}{L_x}$

해상도: $\delta x = \frac{L_x}{2}$

이 경우에는 R_0나 x_c에 대하여 해상도가 변하지 않으며, 단지 안테나 구경의 절

반의 해상도를 가지게 된다. 이는 SAR strip mode의 전형적인 예에 해당하며 안테나 빔폭이 허용하는 길이만큼 충분히 azimuth 방향으로 샘플링이 이루어지는 경우이다. 그러한 의미에서 이 영역의 SAR 영상의 특성을 Full Focusing이라 지칭할 수 있다.

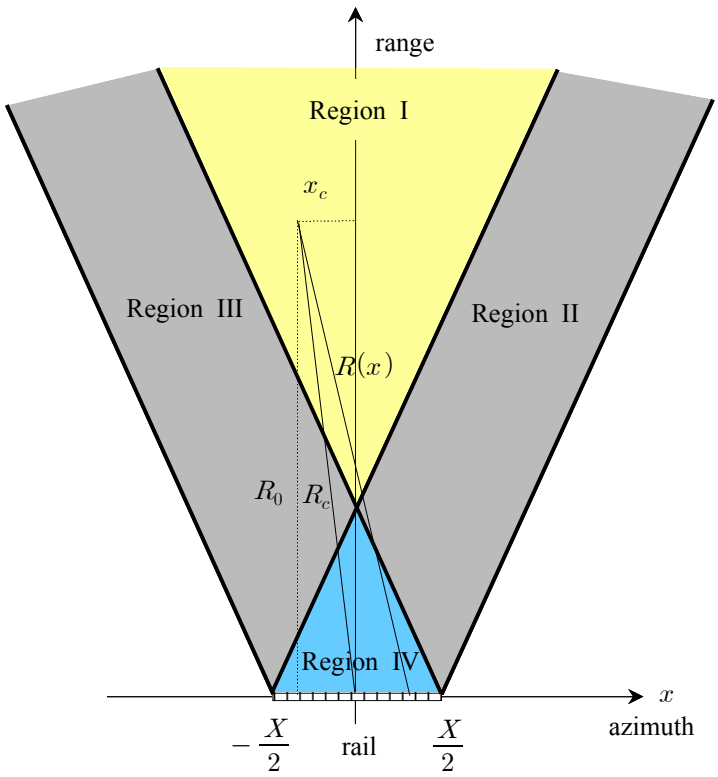


그림 2. GB-SAR geometry and image regions.

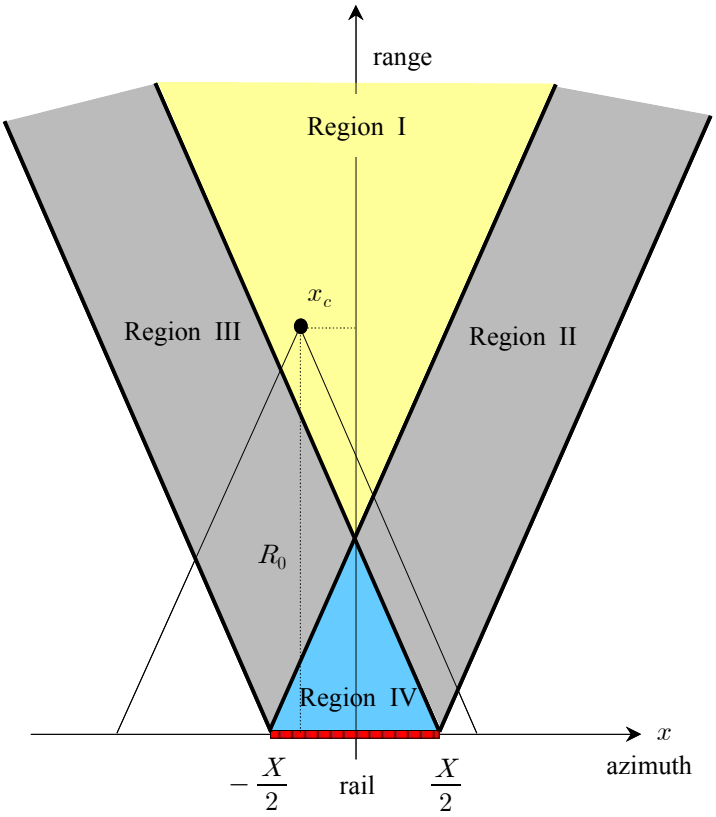


그림 3. Synthetic aperture for a point target in Region I.

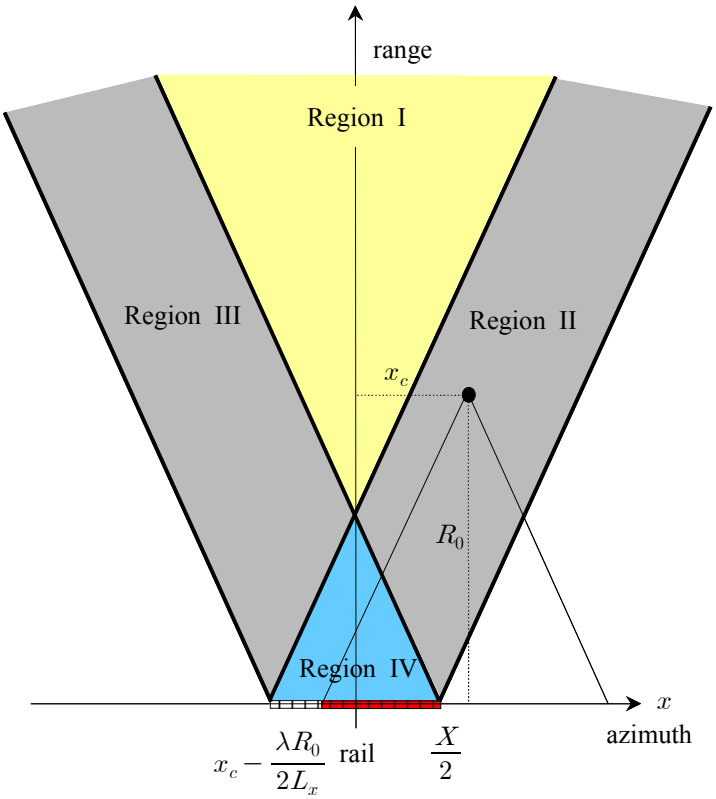


그림 4. Synthetic aperture for a point target in Region II.

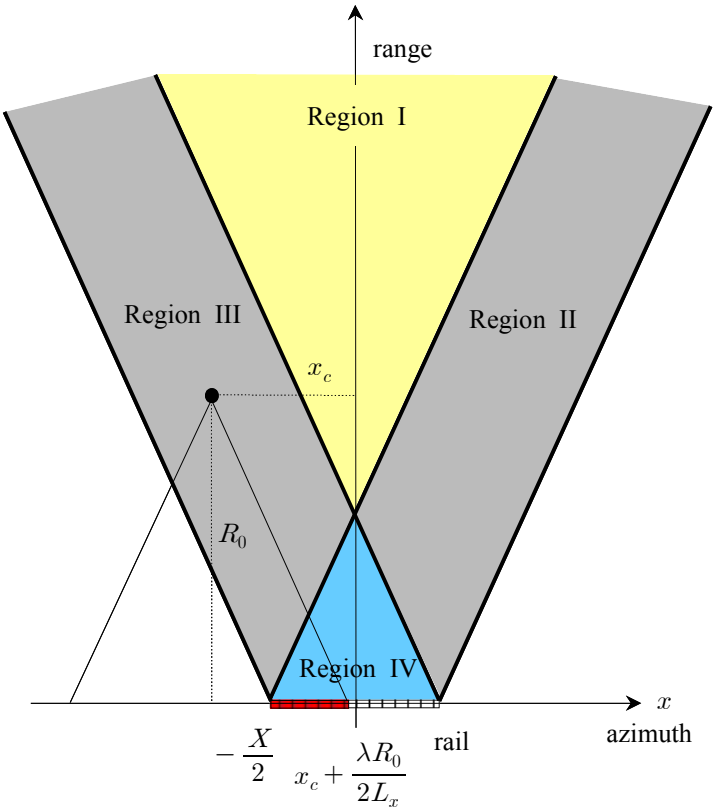


그림 5. Synthetic aperture for a point target in Region III.

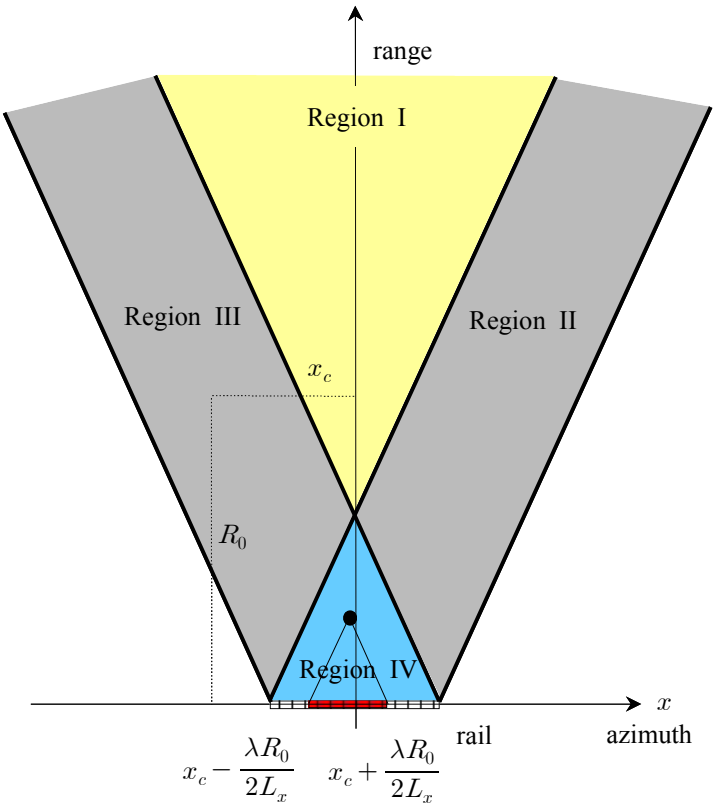


그림 6. Synthetic aperture for a point target in Region IV.

3.3. GB-SAR 간섭 기법 (GB-InSAR)

SAR Focusing 후 영상에서 얻어지는 위상(phase)은 다음과 같이 거리의 함수이다.

$$\phi = -\frac{4\pi}{\lambda} R \tag{45}$$

시간, 공간, 혹은 중심 진동수를 달리하여 얻어진 영상에서 얻어지는 위상의 차이를 이용하여 산란체의 움직임이나 지형을 알아내는 기술을 간섭기법(Interferometry)라 한다. 여기서는 GB-SAR 시스템을 이용하여 구현할 수 있는 여러 interferometry의 가능성을 살펴보고자 한다.

(1) Differential Interferometry (DInSAR) for coherent target-motion detection

같은 조건에서 시간차를 두고 얻어진 서로 다른 GB-SAR영상에서의 위상차(phase difference)를 생각하자. 그림 7과 같이 하나의 산란체가 레이더 방향으로 r 만큼 이동했을 때, 각각의 위상은 다음과 같다.

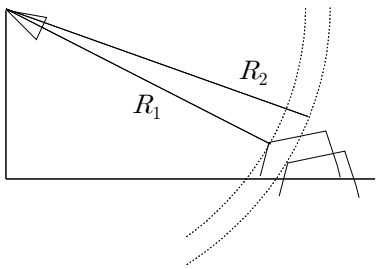


그림 7. DInSAR configuration with target movement

$$\phi_1 = -\frac{4\pi}{\lambda} R_1, \quad \phi_2 = -\frac{4\pi}{\lambda} R_2 = -\frac{4\pi}{\lambda}(R_1 + r)$$

$$\text{phase difference: } \boxed{\phi = \phi_2 - \phi_1 = -\frac{4\pi}{\lambda} r} \tag{46}$$

$$\text{motion sensibility: } \boxed{\frac{\partial \phi}{\partial r} = -\frac{4\pi}{\lambda}} \tag{47}$$

예를 들면, C-band일 경우 $\triangle\phi = -12720\triangle r$ (degree)로서, 산란체가 레이더 방향으로 1mm의 이동만 있어도 12.7도의 위상 변화가 일어난다. 보통 인공위성 및 GB-SAR 시스템의 위상 측정 오차가 10도 미만일 경우, DInSAR의 산란체 이동

에 대한 민감도는 1mm라 할 수 있다.

(2) Cross-track interferometry for DEM (topography)

그림 8과 같이 첫 번째 영상과 수직기선 (vertical baseline)을 두고 얻어진 두 번째 영상간의 위상차를 생각하자.

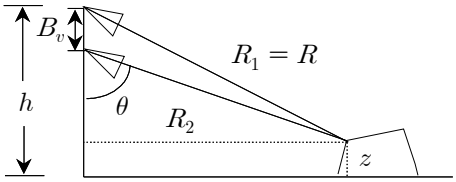


그림 8. InSAR Configuration.

$$\phi_1 = -\frac{4\pi}{\lambda} R_1, \quad \phi_2 = -\frac{4\pi}{\lambda} R_2$$

phase difference: $\phi = \phi_2 - \phi_1 = -\frac{4\pi}{\lambda}(R_2 - R_1) = -\frac{4\pi}{\lambda}\triangle R$

삼각형에서

$$R_2 = \sqrt{R^2 + B_v^2 - 2B_v R\cos\theta} \qquad (R_1 = R \text{ for brevity})$$

$$= R\sqrt{1 + \frac{B_v^2}{R^2} - \frac{2B_v}{R}\cos\theta}$$

$$\simeq R\left(1 - \frac{B_v}{R}\cos\theta\right) \qquad (\because \sqrt{1 \pm x} \simeq 1 \pm \frac{1}{2}x)$$

InSAR phase difference:

$$\boxed{\therefore \phi = \frac{4\pi}{\lambda} B_v \cos\theta = \frac{4\pi h B_v}{\lambda R}} \tag{48}$$

Range fringe ramp:

$$\frac{\partial\phi}{\partial R} = -\frac{4\pi}{\lambda} B_v \sin\theta \frac{\partial\theta}{\partial R},$$

$$h - z = R\cos\theta, \ \frac{\partial\theta}{\partial R} = \frac{1}{R\tan\theta}$$

$$\boxed{\therefore \frac{\partial\phi}{\partial R} = -\frac{4\pi}{\lambda R} B_v \cos\theta = -\frac{4\pi h B_v}{\lambda R^2}} \tag{49}$$

따라서, 2π cycle 변화를 가지는 range fringe의 파장은 $\triangle R_{2\pi} = -\frac{\lambda}{2B_v h} R^2$으로서, R^2에 비례한다. 예를 들어, $R = 50m$, $B_v = 0.3m$, $h = 5m$일 때, $\triangle\phi = 0.133\triangle R$이며, 한 cycle ($\triangle\phi = 2\pi$)의 fringe 변화는 $\triangle R = 47m$에 해당한다.

Height sensitivity:

$$\frac{\partial\phi}{\partial z}=-\frac{4\pi}{\lambda}B_v\sin\theta\frac{\partial\theta}{\partial z}$$

$$h-z=R\cos\theta,\ \ \frac{\partial\theta}{\partial z}=\frac{1}{R\sin\theta}$$

$$\boxed{\therefore\ \frac{\partial\phi}{\partial z}=-\frac{4\pi B_v}{\lambda R}} \tag{50}$$

즉, $\triangle z=-\frac{\lambda R}{4\pi B_v}\triangle\phi$로서, 위와 마찬가지로 $R=50m$, $B_v=0.3m$일 때 $\triangle z=0.75\triangle\phi$이다. 따라서 4.7미터의 높이 차이가 있을 경우 한 사이클(2π) fringe가 나타나게 된다.

(3) $\triangle k$-interferometry for DEM

모든 조건이 동일하나 중심 진동수 $f_2=f_1+\triangle f$만큼 변화시킨 경우를 생각하자.

$$\phi_1=-\frac{4\pi R}{c}f_1,\quad \phi_1=-\frac{4\pi R}{c}f_2$$

Phase difference: $\boxed{\phi_1=-\frac{4\pi R}{c}\triangle f}$ (51)

Range fringe ramp: $\boxed{\frac{\partial\phi}{\partial R}=-\frac{4\pi}{c}\triangle f}$ (52)

예를 들어, $\triangle f=10\mathrm{MHz}$인 경우, $\triangle R=-2.39\triangle\phi$이며, 거리 15.00m마다 2π 사이클이 고정적으로 나타난다.

Height sensitivity:

$$\frac{\partial\phi}{\partial z}=-\frac{4\pi\triangle f}{c}\frac{\partial R}{\partial z},$$

$$h-z=R\cos\theta,\ \ \frac{\partial R}{\partial z}=-\frac{1}{\cos\theta},$$

$$\boxed{\therefore\ \frac{\partial\phi}{\partial z}=\frac{4\pi\triangle f}{c\cos\theta}=\frac{4\pi h\triangle f}{cR}} \tag{53}$$

즉, $R=50m$, $\triangle f=10\mathrm{MHz}$, $h=5m$일 때, $\triangle z=23.87\triangle\phi$가 되며, 고도차가 $\triangle z=150m$ 일 때 2π 사이클이 나타난다.

(4) Cross-track and $\triangle k$-interferometry for DEM

Vertical baseline B_v와 중심 진동수 $\triangle k$가 동시에 존재하면 위상은 다음과 같은 수식이 된다.

$$\phi_1=-\frac{4\pi}{c}f_1R_1,\quad \phi_1=-\frac{4\pi}{c}f_2R_2$$

Phase difference:

$$\phi = -\frac{4\pi}{c}(f_2 R_2 - f_1 R_1)$$

$$= -\frac{4\pi}{c}\left[(f_1 + \triangle f)(R_1 + \triangle R) - f_1 R_1\right]$$

$$= -\frac{4\pi}{c}(f_1 \triangle R + R_1 \triangle f + \triangle f \triangle R)$$

$$\simeq -\frac{4\pi}{c}(f_1 \triangle R + R_1 \triangle f) \quad (\because \triangle f \triangle R \text{ very small})$$

$$\boxed{\therefore \phi = -\frac{4\pi R}{c}\triangle f - \frac{4\pi}{\lambda}\triangle R} \tag{54}$$

Range fringe ramp:

$$\boxed{\frac{\partial \phi}{\partial R} = -\frac{4\pi}{\lambda R}B_v \cos\theta - \frac{4\pi}{c}\triangle f = -\frac{4\pi h B_v}{\lambda R^2} - \frac{4\pi}{c}\triangle f} \tag{55}$$

Height sensitivity:

$$\boxed{\frac{\partial \phi}{\partial z} = -\frac{4\pi B_v}{\lambda R} + \frac{4\pi \triangle f}{c\cos\theta} = -\frac{4\pi B_v}{\lambda R} + \frac{4\pi h \triangle f}{cR}} \tag{56}$$

따라서, cross-track interferometry와 $\triangle k$ interferometry의 효과가 동시에 나타난다.

3.4. GB-SAR 현장 실험

GB-SAR 시스템을 갖춘 후 실내에서 자료의 획득 및 처리를 시도하였다. 그림 9와 같이 알루미늄 사다리 끝에 mesh 형태의 철망 reflector를 설치하였다. 중심에 있는 콘크리트 수조의 방해를 최소화하기 위하여 안테나의 높이와 reflector를 적절하게 위치시켰다. 사진에서 좌에서 우로 레일을 따라 스캔이 이루어졌다. 실내 실험의 구체적인 내용은 다음과 같다.

(1) 실내 실험

가) Data Acquisition

$f = 5.0 \sim 5.6GHz, N_R = 201$

$X = 4m, \Delta x = 0.05m$

나) Range Compression

$B_R = 600MHz$

$$\triangle f = \frac{B_R}{N_R - 1} = \frac{600MHz}{201 - 1} = 3MHz$$

$$\delta R = \frac{2.997925 \times 10^8 m/s}{2 \times 600MHz} = 0.25m$$

$R_{\max} = (201-1) \times 0.25m/2 = 25m$

$N_{Rfft} = 512$

$B_{Rfft} = (N_{Rfft} - 1)\triangle f = (512-1) \times 3MHz = 1.533GHz$

$$\Delta R = \frac{2.997925 \times 10^8 m/s}{2 \times 1.533GMHz} = 0.1m$$

다) Azimuth Compression

$\lambda = 0.0566m$

$N_x = 81$

Region IV: $\delta x = \frac{L_x}{2} = \frac{0.25m}{2} = 0.125m$

$N_{xfft} = 256$

$X_{fft} = (N_{xfft} - 1)\triangle x = (256-1) \times 0.05m = 12.75m$

DF: $\triangle x_{image} = \frac{\lambda R_c}{2X_{fft}} = 0.00222R_c(m)$

DF: $x_{width} = (N_{xfft} - 1)\triangle x_{image} = (256-1) \times 0.00222R_c = 0.566R_c(m)$

RD: $\triangle x_{image} = \triangle x = 0.05m$

RD: $x_{width} = (N_{xfft} - 1)\triangle x_{image} = (256 - 1) \times 0.05m = 12.75m$

그림 10은 Range-Doppler Algorithm을 이용하여 SAR focusing을 한 결과이다. 수평축은 안테나에서 산란체까지의 거리를 나타내며, 수직축은 azimuth로서, 윗부분이 그림 9의 좌측 부분이며 아랫부분은 우측이다. 거리 6.2m에 직선상으로 나타나는 벽에서의 반사 신호 및 거리 5.6m에 있는 reflector에서 오는 신호는 HH에서는 매우 선명하게 나타난 반면, VV에서는 비교적 약한 반사 신호를 보인다. 벽에서 반사된 신호는 콘크리트 수조의 맞은편 벽에서 재반사를 일으켜 multiple reflection의 형태로 약 15m, 23.5m에서 반복해서 나타난다.

영상 전체에 수직선으로 나타나는 잡음 성분 또한 HH보다 VV에서 더 강하다. 단일 안테나로 송수신을 했기 때문에 co-pol에서의 잡음이 cross-pol의 경우보다 매우 강함을 알 수 있다. HV 영상에서는 전체적으로 반사 신호가 약하지만 잡음이 낮아 비교적 뚜렷한 영상이 형성되었다. HV영상에서는 거리 약 4m에서 강한 cross-pol 반사 신호가 잡히는데, 이는 그림 9의 사진에서 전면에 보이는 콘크리트 수조의 중간 부분에 의한 반사로 여겨진다. 한편 그림 11은 Deramp-FFT Algorithm을 이용하여 SAR Focusing을 처리한 결과이다. 이론에서 밝힌 바와 같이 짧은 거리에서 focusing이 잘 이루어지지 않음을 알 수 있다.

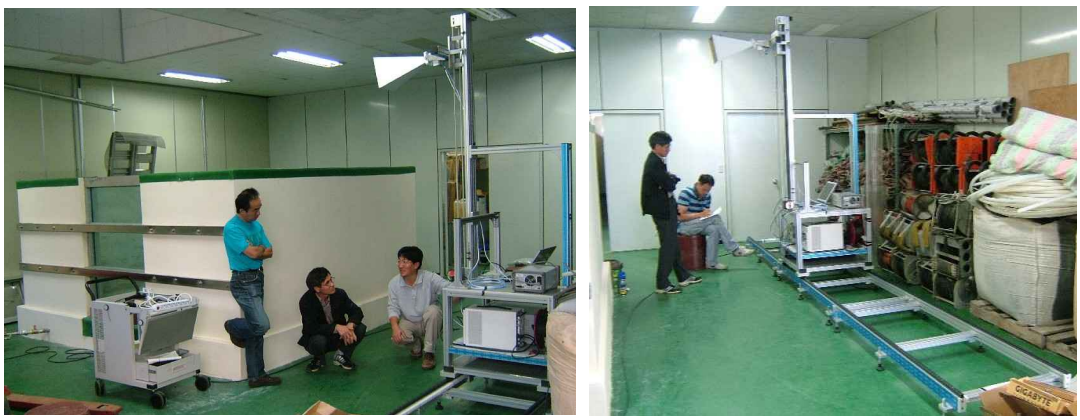
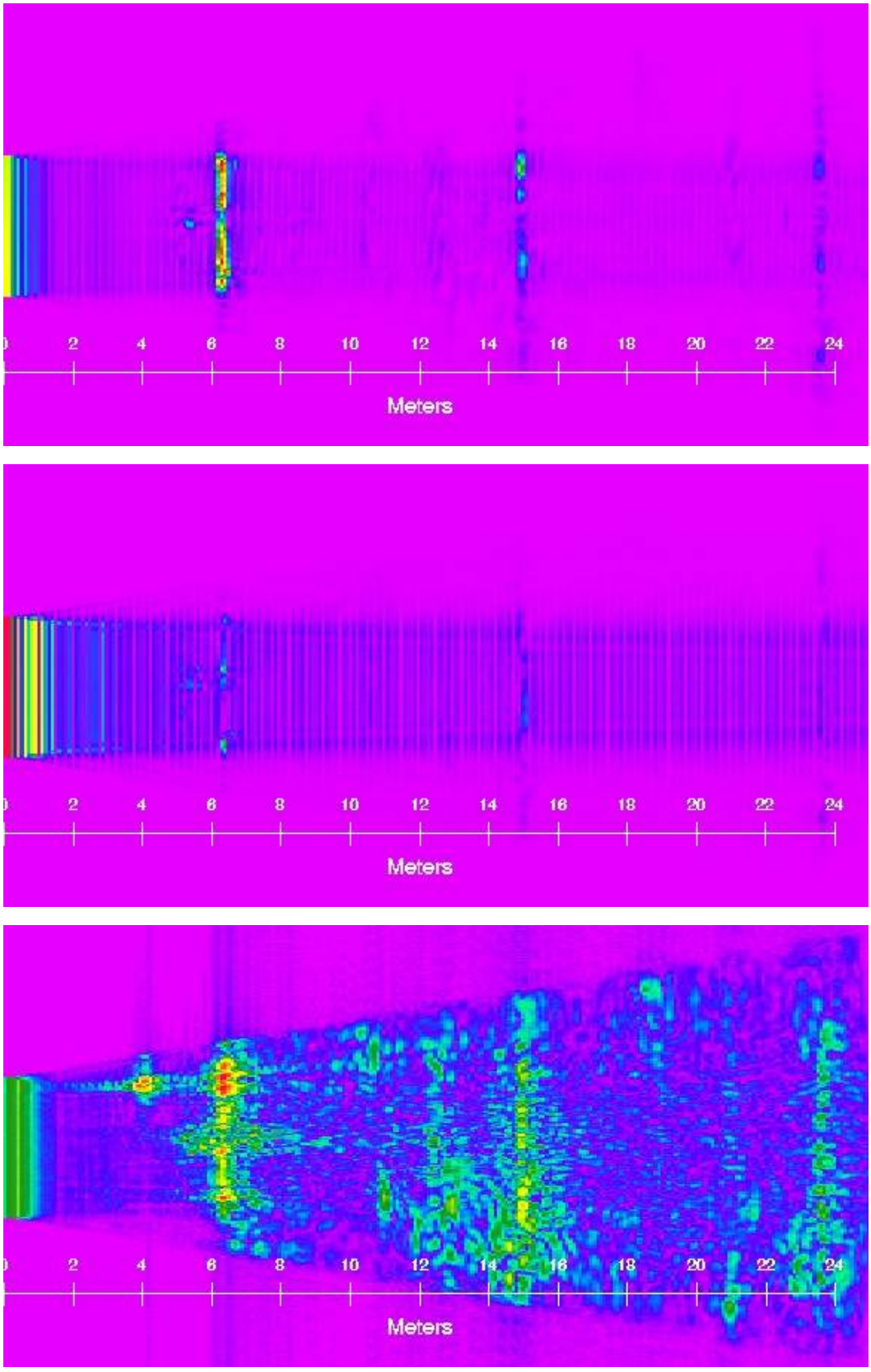


그림 9. Indoor experiment setup. A mesh reflector with aluminium ladder was placed at about 60cm from the wall.



(a) HH (RD)

red: -22dB
yellow: -23dB
green: -24dB
blue: -30dB
purple: -40dB

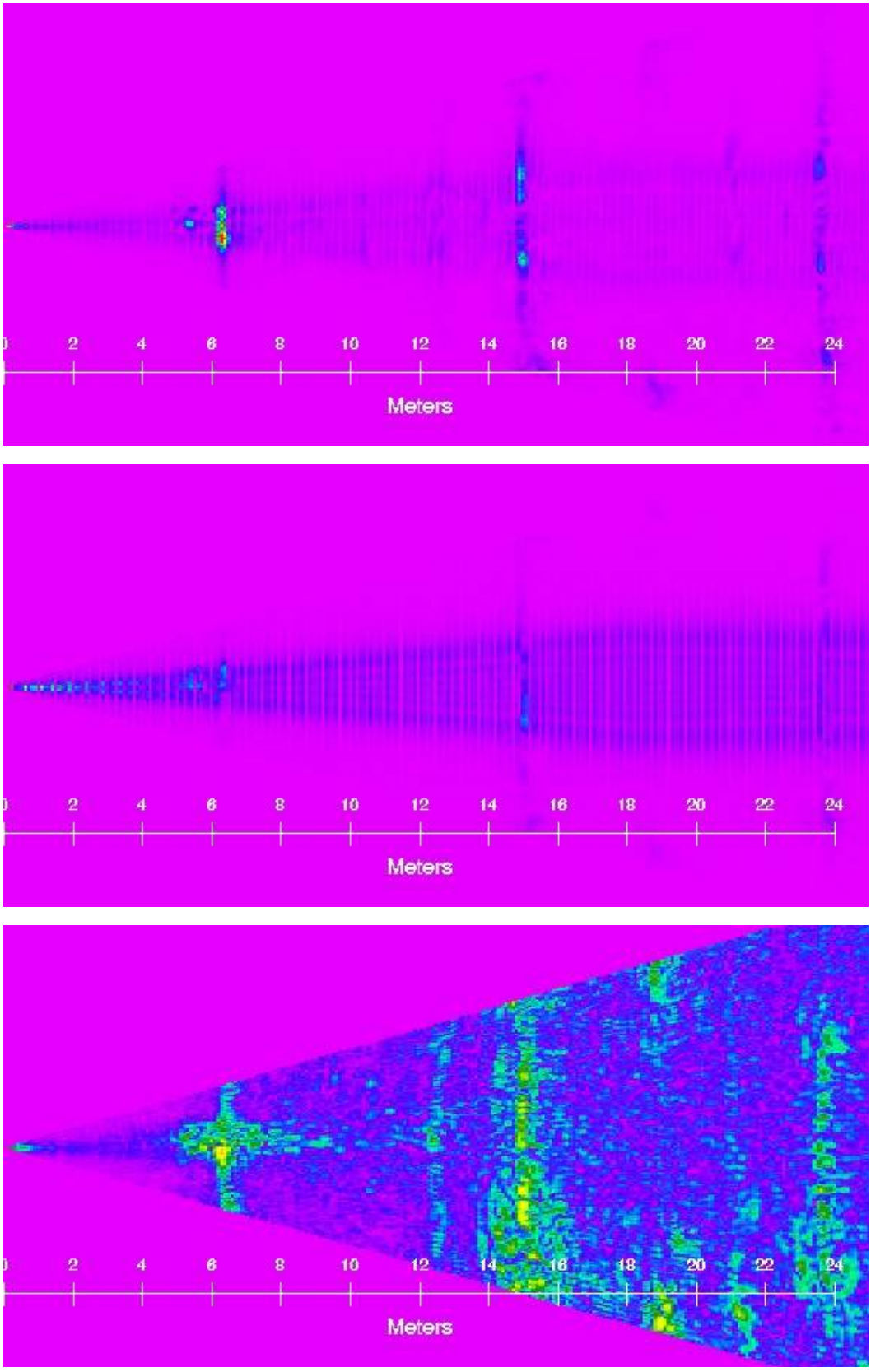
(b) VV (RD)

red: -22dB
yellow: -23dB
green: -24dB
blue: -30dB
purple: -40dB

(c) HV (RD)

red: -48dB
yellow: -60dB
green: -66dB
blue: -74dB
purple: -80dB

그림 10. Indoor Experiment results processed by Range-Doppler algorithm



(a) HH (DF)

red: -22dB
yellow: -23dB
green: -24dB
blue: -30dB
purple: -40dB

(b) VV (DF)

red: -22dB
yellow: -23dB
green: -24dB
blue: -30dB
purple: -40dB

(c) HV (DF)

red: -48dB
yellow: -60dB
green: -66dB
blue: -74dB
purple: -80dB

그림 11. Indoor Experiment results processed by Deramp-FFT Algorithm

(2) 야외 실험

그림 12와 같이 야외에서 corner reflector를 설치하고 GB-SAR 실험을 실시하였다. 안테나로부터 직선 거리로 약 36.09m 지점에 trihedral corner reflector를 설치하였고, 46.38m 지점에 원통형의 mesh reflector를 설치하였다. 주요 target의 수평거리는 다음과 같다.

trihedral corner reflector: 36.09±0.09m
mesh reflector: 46.38m
단풍나무(최단거리): 48.64±0.82m
지자기 관측함(최단거리): 51.41±1.21m
하천제방: 156.32±0.34m
생명공학연구소 빌딩: 253.49m

두 reflector를 잇는 직선은 레일의 중심과 대략적으로 만나며, 레일의 방향과 수직을 이룬다. $f = 5.0 \sim 5.6GHz$, $N_R = 1601$로 하였으며, $X = 5m$, $\Delta x = 0.05m$로 하였다. 실험에 자세한 내역은 Table 2 에 표시하였다. 그림 13은 Range-Doppler Algorithm, 그림 14는 Deramp-FFT Algorithm을 사용하여 SAR focusing한 결과이며, 그림 15는 비교적 선명한 영상을 얻는 HV에 대하여 주요 부분만을 확대한 영상이다.

네트워크 분석기의 최대 파워가 10dBm으로 제한되어 있어 장거리 운용에는 한계가 있었다. 그리고 단일 안테나로 송수신이 이루어졌기 때문에, 네트워크 분석기 내의 directional coupler의 성능 제한으로 말미암아 co-pol (VV, HH)에서는 잡음이 매우 많아 trihedral corner reflector의 신호만이 두드러지게 보이며, 다른 산란체의 모습은 알아볼 수 없었다. 그러나 cross-pol에서는 안테나의 port isolation이 -50dB로 매우 뛰어나기 때문에 잡음이 적어 비교적 선명한 영상을 얻을 수 있었다.

HV 영상에서는 corner reflector, 단풍나무, 관목, 지자기 관측함의 위치가 비교적 정확하게 영상화되었음을 확인할 수 있었다. 또한 약 160m 에서 하천 제방의 모습이 실제와 비슷하게 비스듬한 모습으로 영상화 된 것도 관찰할 수 있었다. 향후 마이크로파 엠프와 마이크로파 스위치를 이용하여 신호대 잡음비를 높여준다면 보다 먼 거리에서도 정확한 영상화가 이루어질 것이며 영상 중심부의 좌우 줄무늬 잡음도 개선될 것으로 기대한다.

Range-Doppler Algorithm은 메모리가 많이 필요한 반면 모든 거리에 대하여 비교적 정확한 영상을 얻었으며, Deramp-FFT Algorithm은 상대적으로 작은 메모리와 CPU 시간으로 Quick Look 영상을 얻는데 적합하지만, 거리가 가까운 곳에서는 영상화가 잘 되지 않는 단점을 잘 보여주었다.

이 실험에서는 GB-SAR Interferometry에 대한 가능성을 알아보기 위하여 다음의 4회에 걸쳐 영상을 얻었다.

가) T1: 사진에서와 같은 target에 대하여 Table 2의 조건으로 영상을 얻었다.

나) T2: 동일 조건에서 trihedral corner reflector의 위치를 2-5mm 앞으로 전진시켜 phase 의 변화를 관찰하고자 하였다. 잔디밭 위에서 손으로 움직이다 보니 이동량을 정확하게 제어할 수는 없었다.

다) T3: 안테나의 위치를 수직으로 30cm 내려서 측정하여, Cross-Track Interferometry를 구현하고자 하였다.

라) T4: T3과 동일 조건에서 중심 주파수를 5.30GHz에서 5.31GHz로, 10MHz를 이동하여 $\triangle K$-interferometry를 구현하였다

DInSAR motion detection (T1-T2)

그림 16(a)는 DInSAR의 가능성을 알아보기 위하여 T1과 동일한 조건에서 trihedral corner reflector의 위치를 약간 변동시켜 T2를 얻고 그 phase 변화를 살펴본 것이다. 전체적으로 신호대 잡음비가 낮은 곳에서는 무작위적인 phase 변화를 보여주고 있으나, corner reflector에서 약간의 위상 변화가 보임을 알 수 있다. 그러나 reflector의 위치를 변화시킬 때 정확한 계측을 할 수 없어서 이를 정량화 할 수는 없었다. 뿐만 아니라, 관목 부분에서도 위상의 변화가 관측되는데, 이는 관목이 바람에 움직였거나 관목의 cross-pol 반사 신호 특성이 매우 불안정하기 때문인 것으로 판단된다. 마찬가지로 시스템의 신호대 잡음비가 개선된다면 보다 장거리에 있는 안정된 산란체의 미세한 움직임을 mm 정밀도로 유추해 낼 수 있을 것으로 기대된다.

Cross-track interferometry for DEM (T2-T3)

그림 16(b)는 T2와 동일한 조건에서 안테나의 위치를 30cm 내려서 T3을 측정하고 그 위상차를 살펴본 것이다. 잡음이 많아서 산란체의 DEM 형태를 쉽게 구별할수는 없었다. 그러나 앞 절에서 계산한 바와 같이 거리에 따른 interferogram의 한 사이클의 주기가 거리의 제곱에 비례함을 알 수 있으며, 그 주기가 이론적으로 계산한 값과 정확하게 들어맞는다. 역시 시스템의 신호대 잡음비가 개선된다면 산란체의 지형 정보를 InSAR를 통해 얻을 수 있을 것으로 기대된다.

$\triangle k$-interferometry for DEM (T3-T4)

그림 16(c)는 T3과 같은 조건에서 단지 중심주파수를 10MHz 증가시킨 상태에서 T4를 얻고 그 위상차를 구한 영상이다. 앞 절의 이론에 의한 range fringe가 거리에 관계없이 일정하며 그 값 또한 이론값과 정확히 일치함을 알 수 있다. 여기서도 interferogram이 높이에 대한 민감도를 가지며, 향후 개선된 시스템에서는 지형의 효과를 볼 수 있을 것으로 기대한다.

Combination of cross-track interferometry and $\triangle k$-interferometry(T2-T4)

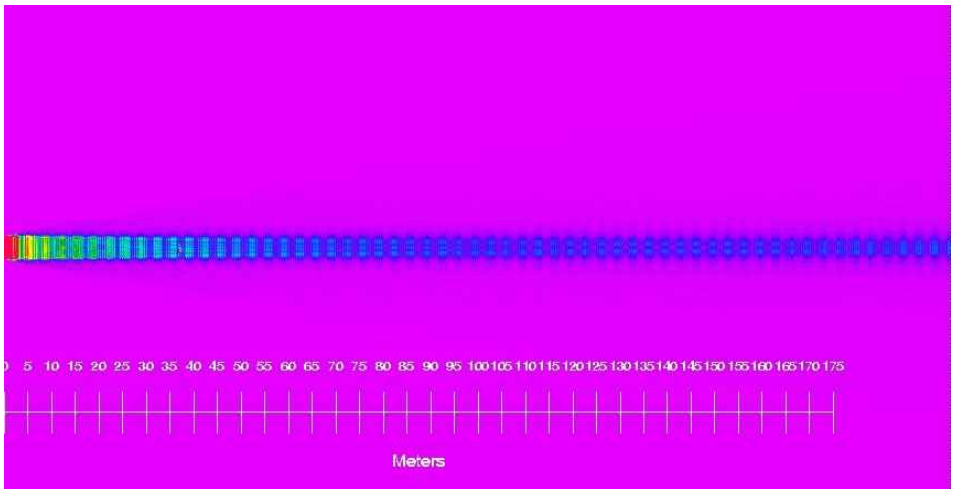
그림 16(d)는 T2에 비하여 30cm의 vertical baseline과 10MHz의 중심주파수 차이를 보이는 T4와의 위상차를 나타낸 영상이다. 여기서는 이론값과 같이 cross-track interferometry와 $\triangle k$-interferometry에 해당하는 range fringe와 height sensitivity가 모두 나타난다. range fringe가 사이클 주기가 이론값과 정확히 일치하는 것으로 볼 때, 실험 결과가 이론에 일치하는 것으로 보이며, 향후 시스템의 신호대 잡음비가 향상되면 지형 고도에 보다 민감한 간섭영상을 얻을 것으로 예측된다.

Table 2. GB-SAR Acronyms, Equations, and Examples. (RD: Range-Doppler algorithm, DF: Deramp-FFT algorithm)

	Symbol	Description	Dimension	Examples
Coordinates	f	frequency	Hz	
	t	time	s	
	R	range	m	
	x	azimuth	m	
	u	wavenumber	1/m	
System	f_c	center frequency	Hz	5.3GHz
	B_R	range bandwidth	Hz	600MHz
	N_R	number of range samples		1601
	X	azimuth(rail) scan length	m	5000mm
	$\triangle x$	azimuth scan step	m	50mm
	L_R	antenna aperture range size	m	25cm
	L_x	antenna aperture azimuth size	m	25cm
	c	speed of light,	m/s	299792500m/s
	λ	wavelength, $\lambda = c/f_c$	m	5.66cm
	θ_R	range beam angle, $\theta_R = \lambda/L_R$	radian	nominal 15deg
	θ_x	azimuth beam angle, $\theta_x = \lambda/L_x$	radian	nominal 15deg
Range Compression	δt	time resolution $\delta t = 1/B_R$	s	1.67ns
	δR	range resolution $\delta R = \frac{c\delta t}{2} = \frac{c}{2B_R}$	m	25cm
	$\triangle f$	frequency step, $\Delta f = \frac{B_R}{N_R - 1}$	Hz	375KHz
	$t_{\max}$	Maximum time, $t_{\max} = \frac{1}{2\Delta f}$ (halved due to void negative time)	s	$1.33\mu s$
	$R_{\max}$	Maximum range, $R_{\max} = \frac{ct_{\max}}{2}$	1/m	199.86m
	N_{Rfft}	number of range samples for FFT, $N_{Rfft} = 2^n \geq N_R$		2048
	B_{Rfft}	range bandwidth for FFT, $B_{Rfft} = (N_{Rfft} - 1)\Delta f$	Hz	768MHz
	Δt	time pixel size, $\Delta t = \frac{1}{B_{Rfft}}$	s	1.30ns
	ΔR	range image pixel size, $\Delta R = \frac{c\Delta t}{2} = \frac{c}{2B_{Rfft}}$	m	19.5cm
Azimuth Compression	N_x	Number of azimuth samples, $N_x = \frac{X}{\Delta x} + 1$		101
	N_{xfft}	Number of azimuth samples for FFT, $N_{xfft} = 2^n \geq N_x$		DF:256, RD:2048
	X_{fft}	Azimuth scan pseudo-length for FFT, $X_{fft} = (N_{xfft} - 1)\Delta x$	m	DF:12.75m, RD: 102.4m

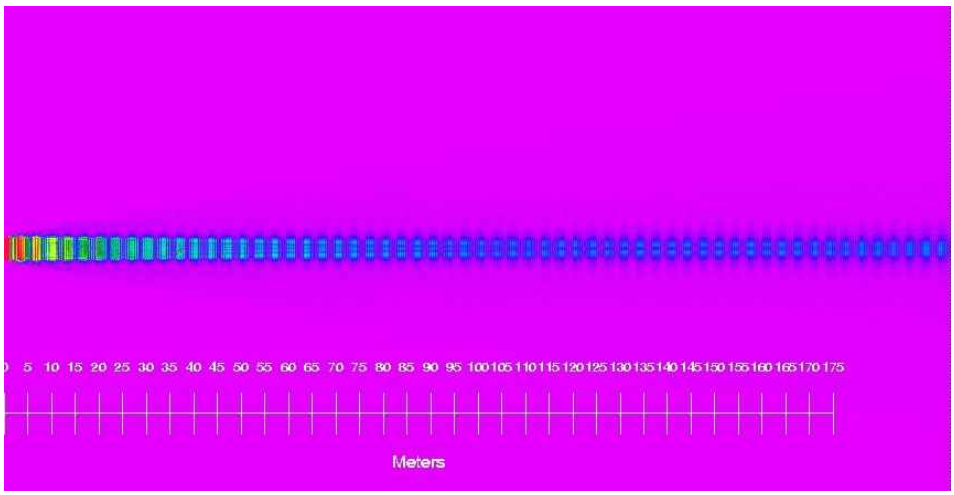
Azimuth Compression (cont')	Δu	Wavenumber pixel size for FFT, $\Delta u = \frac{1}{X_{fft}}$	1/m	DF: 7.843e-2, RD: 9.766e-3
	R_0	shortest range from the extended rail line to target	m	
	R_c	range from aperture center to target, $R_c = \sqrt{R_0^2 + x_c^2}$	m	
	u_{Dc}	wavenumber Doppler centroid	1/m	RD only: 0
	u_R	wavenumber Doppler rate RD: $u_R = -\frac{2}{\lambda R_0}$	1/m/m	RD only
	L_{beam}	beam width, $L_{beam} = R_0 \frac{\lambda}{L_x}$	m	$0.226 R_0$ (m)
	L_s	synthetic aperture length, $L_s \le X$ and $L_x \le L_{beam}$	m	x_c, R_0 dependent
	δx	azimuth resolution DF $\delta x = \frac{\lambda}{2}\frac{R_c}{L_s}$, RD: $\delta x = \frac{\lambda}{2}\frac{R_0}{L_s}$	m	Full focusing: $L_s = L_{beam}$, Partial focusing: $L_s < L_{beam}$
	ϕ	InSAR phase, DF: $\phi = -\frac{4\pi}{\lambda} R_c$, RD: $\phi = -\frac{4\pi}{\lambda} R_0$	radian	Range dependent
	$\triangle x_{image}$	azimuth pixel size of GB-SAR focused image, DF: $\triangle x_{image} = \frac{\lambda R_c}{2}\triangle u = \frac{\lambda R_c}{2X_{fft}}$ RD: $\triangle x_{image} = \triangle x$	m	DF: $0.222 R_c$(cm) RD: 5cm
	x_{width}	azimuth width of GB-SAR focused image, $x_{width} = (N_{xfft} - 1)\triangle x_{image}$ DF: $x_{width} = \frac{\lambda R_c}{2\triangle x}$, RD: $x_{width} = (N_{xfft} - 1)\triangle x$	m	DF: $0.566 R_c$(m) RD: 102.4m

그림 12. Outdoor experiment and panoramic view of target field



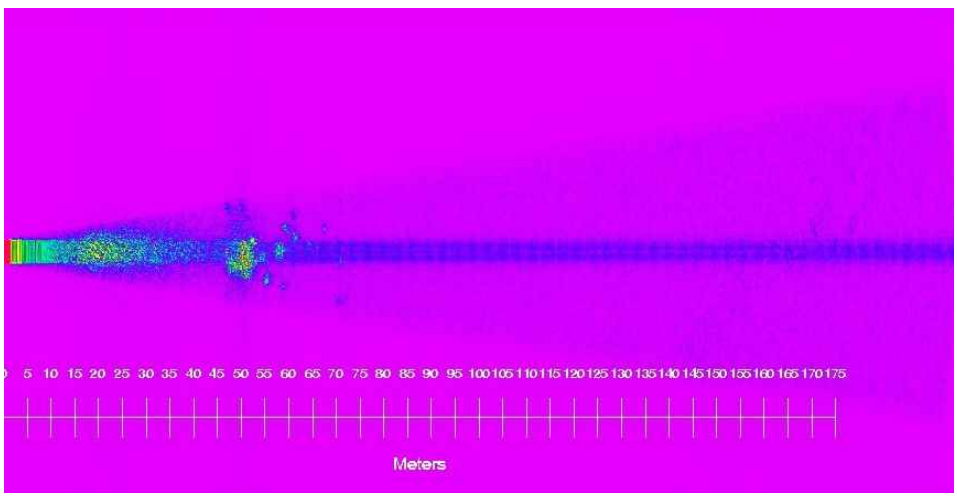
(a) HH (RD)

red: -40dB
yellow: -44dB
green: -46dB
blue: -54dB
purple: -60dB



(b) VV (RD)

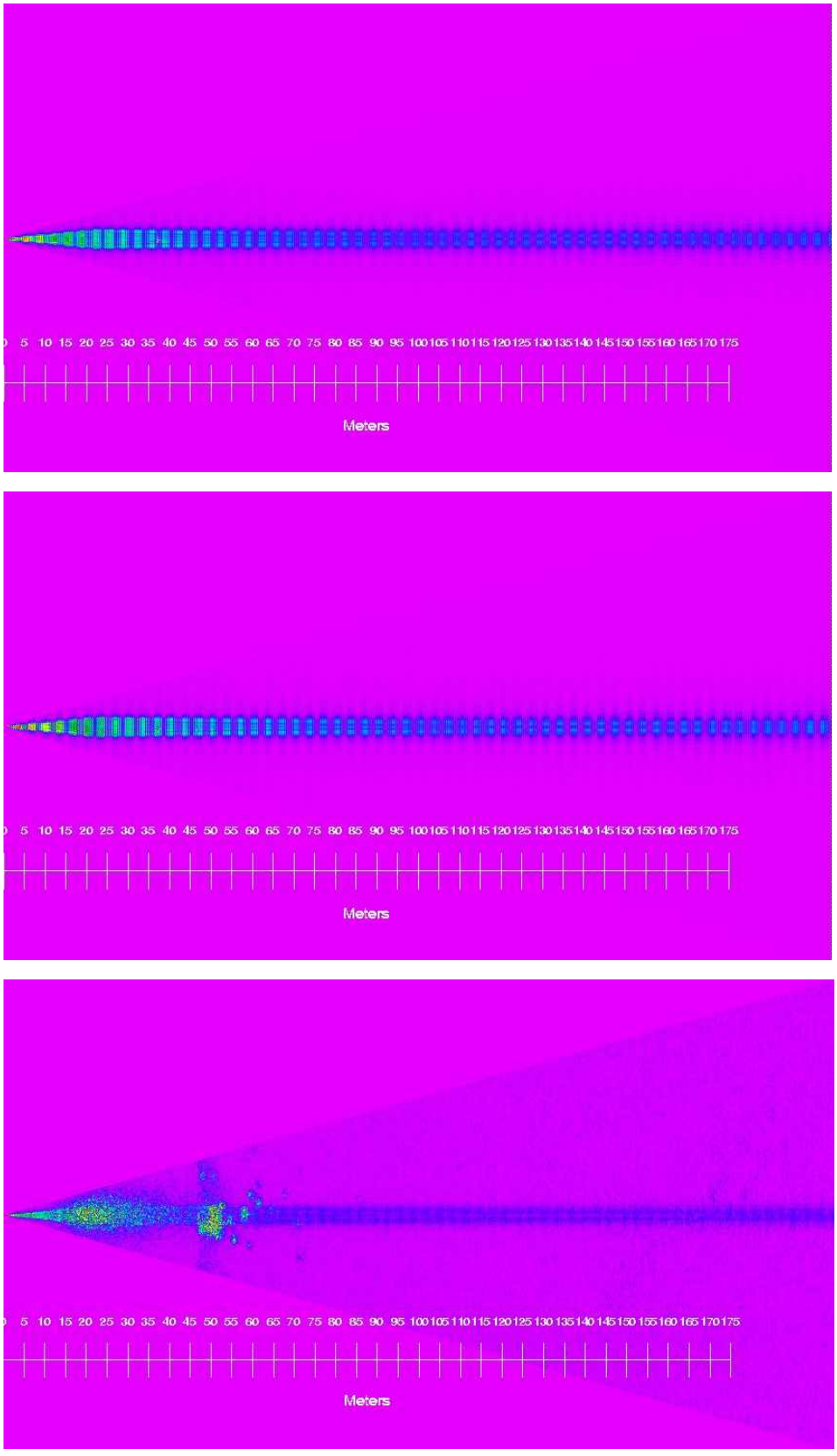
red: -40dB
yellow: -44dB
green: -46dB
blue: -54dB
purple: -60dB



(c) HV (RD)

red: -74dB
yellow: -77dB
green: -80dB
blue: -88dB
purple: -100dB

그림 13. Outdoor experiment results processed by Range-Doppler Algorithm



(a) HH (DF)

red: -40dB
yellow: -44dB
green: -46dB
blue: -54dB
purple: -60dB

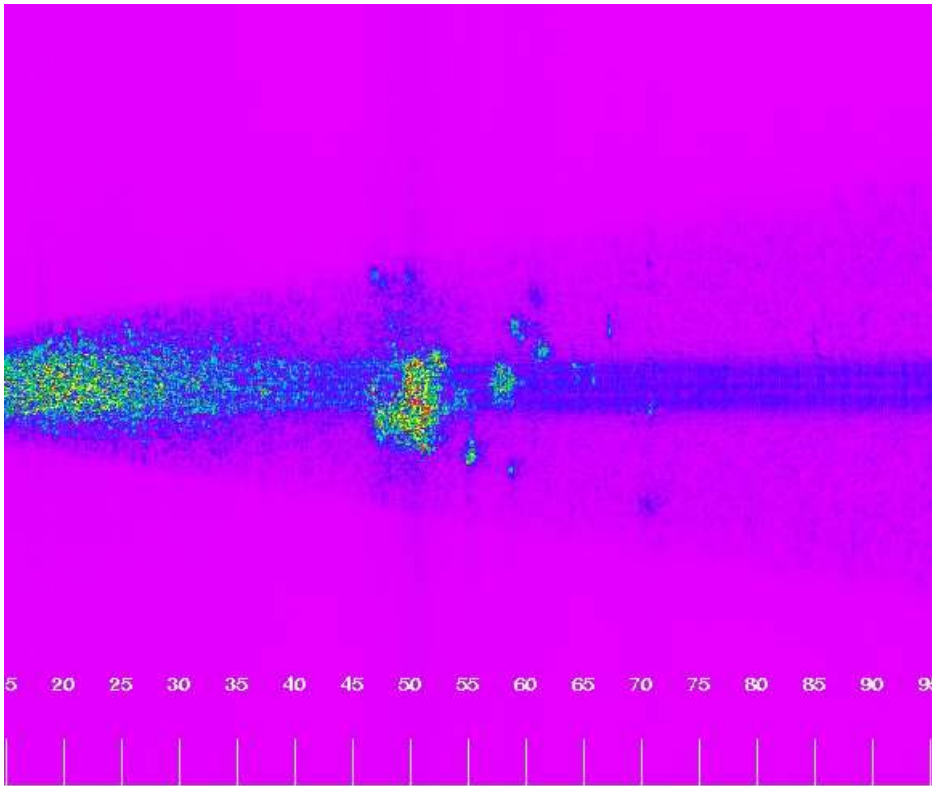
(b) VV (DF)

red: -40dB
yellow: -44dB
green: -46dB
blue: -54dB
purple: -60dB

(c) HV (DF)

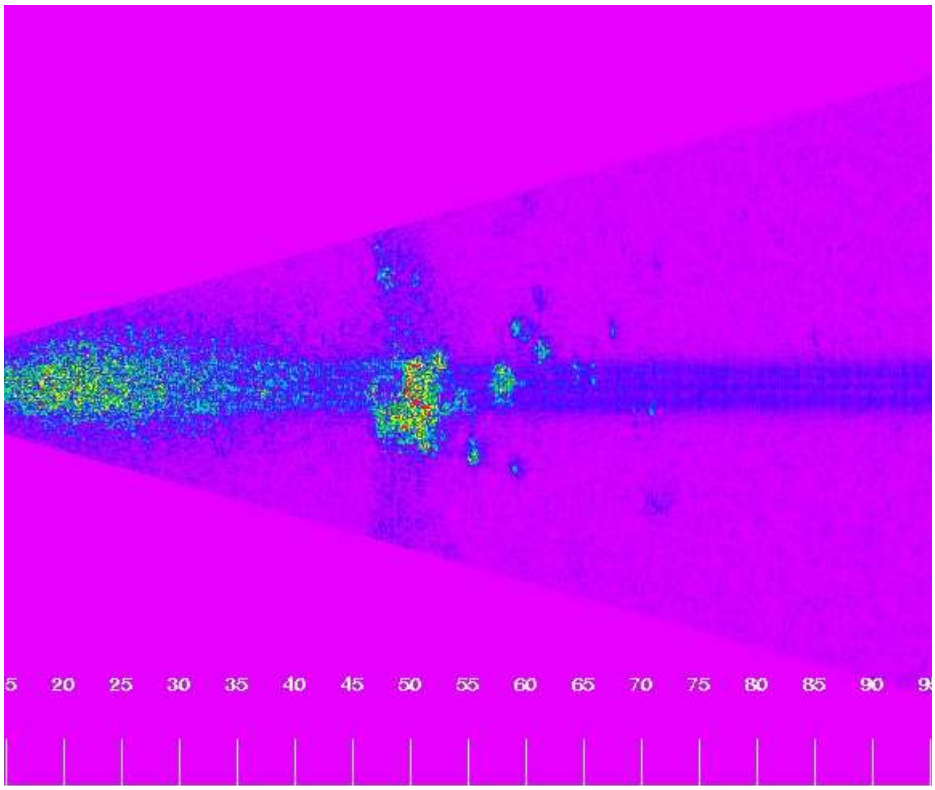
red: -74dB
yellow: -77dB
green: -80dB
blue: -88dB
purple: -100dB

그림 14. Outdoor experiment results processed by Deramp-FFT Algorithm



(a) HV (RD)

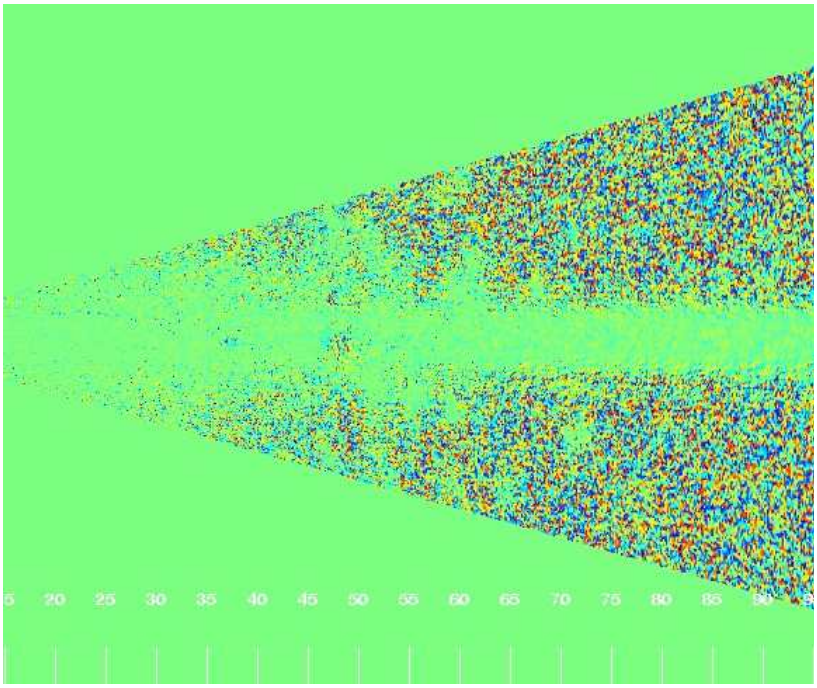
red: -74dB
yellow: -77dB
green: -80dB
blue: -88dB
purple: -100dB



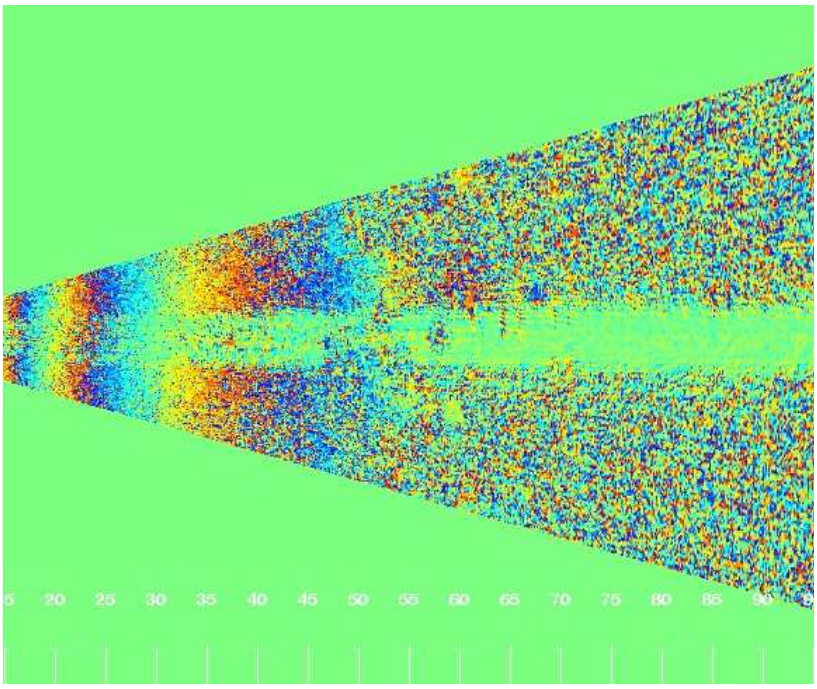
(b) HV (DF)

red: -74dB
yellow: -77dB
green: -80dB
blue: -88dB
purple: -100dB

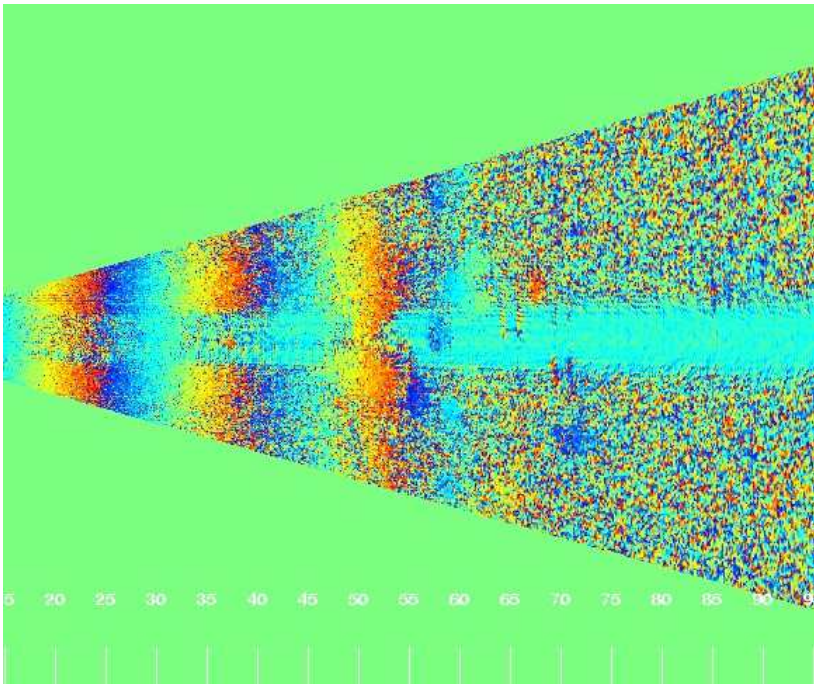
그림 15. A zoomed image of HV processed by (a)RD and (b)DF showing clear target detection.



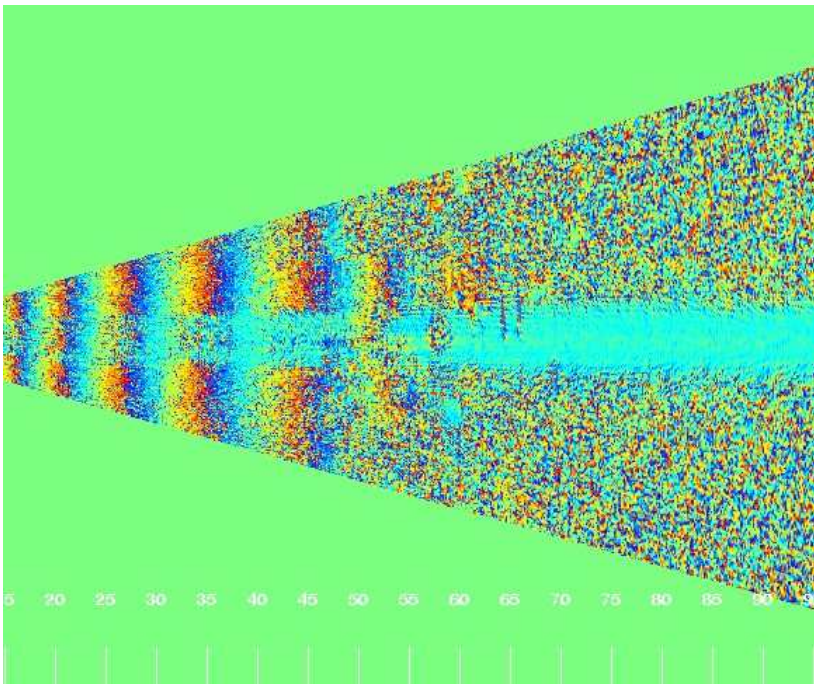
(a) DInSAR (T1−T2) with a shift of trihedral corner reflector by 3mm



(b) Cross−Track InSAR (T2−T3) with 30cm vertical baseline



(c) delta-K InSAR (T3-T4) with 10MHz frequency difference



(d) Cross-Track and $\triangle k$-interferometry (T2-T4) with 30cm vertical baseline and 10MHz frequency difference

그림 16. Possibilities of various GB-SAR Interferometry. (blue to red: $-\pi$ to π)

3.5. 결론

GB-SAR를 구성하고 Range-Doppler Algorithm과 Deramp-FFT Algorithm을 이용한 Focusing 프로그램을 개발하여 실험한 결과, 이론적인 SAR의 최대 해상도를 구현할 수 있었다. 또한 GB-SAR 시스템 운용의 유연성에 기초하여 DInSAR, Cross-track interferometry 및 $\triangle k$-interferometry의 가능성을 확인할 수 있었다. 향후 송신 안테나와 수신안테나를 각기 다르게 하여 신호 잡음을 줄이고, 마이크로파 앰프 및 마이크로파 스위치를 이용하여 '다편광 장거리 GB-SAR'로의 발전을 기대해 본다. 이를 통하여 지반침하 및 산사태로 인한 산란체의 움직임을 장기 관측하는데 활용될 수 있기를 바란다.

제 4 장 목표달성도 및 관련분야에의 기여도

이 연구를 통하여 국내 최초로 GB-SAR 시스템을 구성하고 운용하였으며, 실용화 단계를 눈앞에 두고 있다. GB-SAR 시스템의 안정적인 위상 정보를 이용하면 미약한 산란체의 움직임을 장기간 모니터링하는데 기여할 것으로 기대된다. 또한 이번 연구로 구축된 GB-SAR 시스템은 앞으로 항공기 및 인공위성 SAR 시스템의 개념 설계를 위하여 유용하게 이용될 수도 있다.

제 5 장 연구개발결과의 활용계획

GB-SAR 시스템은 인공위성 SAR의 영상기법에 의거하여 2차원적인 고해상도 영상을 지상에서 얻을 수 있다. 이를 이용하면 마이크로파와 산란체간의 다양한 물리적 현상을 현장에서 규명할 수 있고, 지반침하, 사면 안정성, 산사태, 눈사태 등 다양한 재해 방지에 활용이 높을 것으로 전망된다. 향후 활용을 위하여 댐, 사면, 침하지역, 눈, 건물과 같은 구조물에 대한 안정성 평가에 있어서 새로운 기술로서 활용할 계획이다.

GB-SAR 시스템의 안정적인 위상 정보를 이용하면 미약한 산란체의 움직임을 장기간 모니터링하는데 기여할 것으로 기대된다. 또한 이번 연구로 구축된 GB-SAR 시스템은 국내에서 최초로 시도되는 것으로서, 앞으로 항공기 및 인공위성 SAR 시스템의 개념 설계를 위하여 이용될 수도 있다.

제 6 장 위탁연구과정에서 수집한 해외과학기술정보

본 연구에서는 안테나 하나를 이용하여 마이크로파를 CW 방식으로 송신함과 동시에 수신을 하고 있기 때문에 co-pol에서 잡음이 많이 첨가된다. 이러한 잡음을 없애기 위해서는 수신 안테나와 송신 안테나를 따로 두는 dual-antenna 모드를 구성할 수 있다.

Agilent 사의 PNA Microwave Network Analyzer 시리즈 중에서 Pulsed-RF Measurement Option을 선택하면 single antenna configuration의 약점인 co-pol에서 약 10dB 이상 노이즈 감소 효과가 있기 때문에(Agilent Technologies, 2005; Liepa et. al, 1989), 측정 거리를 백여 미터로 멀리 둘 수 있을 것이며, 현장 실험 구성에 많은 도움이 되겠다.

마이크로파 RF 레이더 시스템에서 안테나의 역할은 매우 중요하다. 좋은 안테나 특성을 얻기 위한 많은 연구가 진행되고 있으며, 국내외 전문 회사도 다양하여 향후 다양한 산란계 및 복사계 시스템을 구성할 수 있다. 이번 실험에서 사용된 Orthogonal Transducer를 이용한 Dual-Polarization Horn Antenna 역시 안테나 특성이 매우 좋은 것으로, 제작 회사의 실험 결과 나타났다. 실험에 사용할 수 있는 안테나의 종류도 매우 다양하여, 1-12GHz, 혹은 1-18GHz에 이르는 광대역 Horn Antenna의 확보도 용이할 것으로 판단된다. 이 외에도 실험 목적에 따라 여러 가지 다양한 안테나를 선택적으로 주문 제작 및 구입할 수 있는 넓은 시장이 존재한다.

현재의 GB-SAR 시스템은 네트워크 분석기의 송신 파워 제한(10dBm)으로 약 100미터 이상 거리에 있는 산란체를 영상화하지 못한다. 따라서 마이크로파 앰프 및 스위치를 이용하여 파워를 높일 필요가 있다. 그리고 송신 및 수신 안테나를 분리하여 잡음을 없애줄 필요가 있다.

제 7 장 참고문헌

이훈열, 2005. 교육용 합성구경레이더 프로세서(eSAR Processor)의 개발과 공개, 대한원격탐사학회지, 21권, 2호, pp. 163-171.

Agilent Technologies, 2005. Agilent PNA Microwave Network Analyzers for Pulsed-RF Measurements - Configuration Guide. www.agilent.com.

Hamasaki, T. Zhou, Z., and Sato, M., 2004. Development and Applications of an Interferometric Ground-Based SAR System, http://earth.esa.int/workshops/ ceos_sar_2004/, accessed on 12 Oct. 2005.

Lee, H. and Chi, K. H., 2004. Contraction of a newly reclaimed mudflat detected by differential SAR interferometry, International Symposium on Remote Sensing (ISRS) 2004, Jeju, Korea.

Lee, H. and J. G. Liu, 2001. Analysis of Topographic Decorrelation in SAR Interferometry using Ratio Coherence Imagery, IEEE Trans. Geosci. Remote Sens., 39(2), 223-232.

Liepa, V. V., Sarabandi, K., and Tassoudji, M. A., 1989. A Pulsed Network Analyzer Based Scatterometer, IGARSS'89, pp. 1826-1828.

SCEOS, 2005. Ground-based Synthetic Aperture Radar (GB-SAR) Research, http://www.shef.ac.uk/~sceos/environmental/gbsar/ , accessed on 12 Oct 2005.

Ulaby, F. T. and Whitt, M. W., 1990. AVNA-Based Polarimetric Scatterometers, IEEE Transactions on Geoscience and Remote Sensing, vol. 28, no. 1, pp. 6-17.

Appendix A. GB-SAR 프로그램 원시코드

< Main Program: gbsar.c >

```
/*****************************************************

        GB-SAR Processor

Input: complex data in frequency domain
Process:
        1. Parameter Control
        2. Range Compression by iFFT
        3. Azimuth Compression
          - Partial Focusing: Deramp-FFT Algorithm
          - Full Focusing: Range-Doppler Algorithm
Output: SAR focused data in time domain

by Hoonyol Lee,
Department of Geophysics
Kangwon National University
Email: hoonyol@kangwon.ac.kr

Versions
1.0 (19 Oct 2006): Initial release
1.1 (11 Dec 2006): Big fix in RD
*****************************************************/
#define Version "1.1"

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include <time.h>

#include "ma.c"
#include "cnshift.c"
#include "cfft.c"
#include "cfout.c"
#include "gen_ers.c"

void usage();
void intro();

main(int argc, char *argv[])
{
        FILE *fp;
        char fname[200], fhead[200], fheadout[200], dummy[200];
        char fvv[200], fvh[200], fhv[200], fhh[200];
        char foutvv[200], foutvh[200], fouthv[200], fouthh[200];
        int i, ix, ix1, ix2, j, jx, jx1, ii;
        int ystart, yend, ystep;
        int nrange, nazimuth, nrange_fft, nazimuth_fft, nrange_ffth;
        int ifocus, iRx, mfft_range, mfft_azimuth;
        time_t clock_start, clock_end, clock_duration;
        double fstart, fend, bandw_r, fc, lambda;
        double delf, delx;
        double delt, delt_fft, delr, delr_fft, m_total, t_max, R_max, R_full, R_center;
        double delu, delu_fft, delx_fft, delx_fftr;
        double cspeed=2.997925e8;
        double phi=4.*atan(1.);
        double freq, *vv, *vh, *hv, *hh;
        double L_az, beam_angle, X, X_fft, X_rd;
        double x, R, R0, Rc, Rx, phase, phase0, phase1, tmp1, tmp2;
        double X_hcut, cosp, sinp;
        double *vvr, *vhr, *hvr, *hhr; /* Range-Doppler Algorithm */
        double *azvv, *azvh, *azhv, *azhh, *azchirp, u;

        intro();
        if(argc !=2) {usage(); exit(0);}

        clock_start=time(NULL);

        /* Parameter file input */
        if((fp=fopen(argv[1], "r"))==NULL)
        {
                printf("Error.. can not open %s.₩n", argv[1]);
                exit(0);
        }
        fscanf(fp, "%s %s", fhead, dummy);
        fscanf(fp, "%s %s", fheadout, dummy);
        fscanf(fp, "%lf %lf %d %s", &fstart, &fend, &nrange, dummy);
        fscanf(fp, "%lf %s", &L_az, dummy);
        fscanf(fp, "%d %d %d %s", &ystart, &yend, &ystep, dummy);
        fscanf(fp, "%d %d %s", &mfft_range, &mfft_azimuth, dummy);
        fscanf(fp, "%lf %s", &R_center, dummy);
```

```
        fscanf(fp, "%d %s", &ifocus, dummy);

        /* output file names */
        strcpy(fvv, fheadout);
        strcpy(fvh, fheadout);
        strcpy(fhv, fheadout);
        strcpy(fhh, fheadout);

        strcat(fvv, "_VV");
        strcat(fvh, "_VH");
        strcat(fhv, "_HV");
        strcat(fhh, "_HH");

        /* Range parameter Input */
        printf("<< Range Parameters >>₩n");
/*      printf(" Enter start, end frequency(GHz) and number of sample: ");
        scanf("%lf %lf %d", &fstart, &fend, &nrange);
*/      fstart *= 1.e9;
        fend *= 1.e9;
        printf("  fstart=%gHz, fend=%gHz nrange=%d₩n", fstart, fend, nrange);
        printf("  FFT folds in Rangge and Azimuth = %d %d₩n",
                  mfft_range, mfft_azimuth);

        fc=(fstart+fend)/2.;
        lambda=cspeed/fc;
        printf("  Center frequency= %gHz₩n", fc);
        printf("  Wavelength= %gm₩n", lambda);

        /* Range Induced Parameters */
        bandw_r=fend-fstart;
        delt=1./bandw_r;
        delr=cspeed*delt/2.;
        delf=bandw_r/(nrange-1);
        t_max=1./(2.*delf);
        R_max= cspeed*t_max/2.;
        nrange_fft=1;
        while(nrange_fft<nrange) nrange_fft *=2;
        nrange_fft *=mfft_range;
        delt_fft=1./((nrange_fft-1)*delf);
        delr_fft=cspeed*delt_fft/2.;
        nrange_ffth=nrange_fft/2;

        printf("  Frequency bandwidth (bandw_r)= %gHz₩n", bandw_r);
        printf("  Time resolution (delt)= %gs₩n", delt);
        printf("  Range resolution (delr)= %gm₩n", delr);
        printf("  Frequency resolution (delf)= %gHz₩n", delf);
        printf("  Time maximum(t_max)= %gs₩n", t_max);
        printf("  Range maximum(R_max)= %gm₩n", R_max);
        printf("  Frequency iFFT pixel size (nrange_fft)= %d₩n", nrange_fft);
        printf("  Time sampling resolution (delt_fft)= %gs₩n", delt_fft);
        printf("  Range sampling pixel size (delr_fft)= %gm₩n", delr_fft);
        printf("₩n");

        /* Azimuth parameter input */
        printf("<< Azimuth parameters >>₩n");
/*      printf(" Enter azimuth antenna length(L_az) (0.25): ");
        scanf("%lf", &L_az);
*/      printf("  L_az = %gm. The ystep should be less than %dmm.₩n",
                  L_az, (int)(L_az*1000/2));
/*      printf(" Enter ystart, yend, ystep (mm, integer): ");
        scanf("%d %d %d", &ystart, &yend, &ystep);
*/      printf(" ystart= %dmm, yend= %dmm, ystep= %dmm₩n",
                  ystart, yend, ystep);

        /* Azimuth induced parameters */
        X = (double)(yend-ystart)/1000.;
        delu=1./X;
        R_full=L_az/lambda*X;
        nazimuth=(yend-ystart)/ystep+1;
        delx = ystep/1000.;
        nazimuth_fft=1;
        while(nazimuth_fft<nazimuth) nazimuth_fft *=2;
        nazimuth_fft *=mfft_azimuth;
        X_fft =(nazimuth_fft-1)*delx;
        delu_fft=1./X_fft;
        delx_fft=lambda/2.*delu_fft;

        printf(" Rail scan length(X)= %gm₩n", X);
        printf(" Azimuth wavenumber resolution(delu)= %g[1/m]₩n", delu);
        printf(" Full focusing Range limit (R_full)=%gm₩n", R_full);
        printf(" Number of azimuth samples (nazimuth)= %d₩n", nazimuth);
        printf(" Azimuth sampling step (dela)= %gm₩n", delx);
        printf(" Azimuth FFT size (nazimuth_fft)= %d₩n", nazimuth_fft);
        printf(" Rail scan length for FFT (X_fft)= %gm₩n", X_fft);
        printf(" Azimuth wavenumber pixel size(delu_fft)= %g[1/m]₩n", delu_fft);

        if(R_center==0.) R_center=R_max;
        delx_fft *= R_center;
        printf(" Azimuth pixel size =%gm at R=%gm.₩n", delx_fft, R_center);
        printf("₩n");
```

```
	/* GB-SAR Algorithm Selection */
/*	printf("<< GB-SAR Algorithm Selection >>₩n");
	printf("  Far/Short Range Criterion=%gm₩n", R_full);
	printf("  1. Deramp-FFT Algorithm (Far Range)₩n");
	printf("  2. Range-Doppler Algorithm (Short Range)₩n");
	printf("  3. Both₩n");
	printf(" Enter the choice: ");
	scanf("%d", &ifocus);
*/	if(ifocus==1) printf("Deramp-FFT Algorithm₩n");
	if(ifocus==2) printf("Range-Doppler Algorithm₩n");
	else if(ifocus==3) printf("Deramp-FFT and Range-Doppler Algorithms₩n");
	else
	{
		printf("  Error.. wrong input₩n");
		exit(0);
	}
	printf("₩n");

	/* memory allocation */
	vv=dvector(nrange_fft*nazimuth_fft*2);
	vh=dvector(nrange_fft*nazimuth_fft*2);
	hv=dvector(nrange_fft*nazimuth_fft*2);
	hh=dvector(nrange_fft*nazimuth_fft*2);

	m_total=nrange_fft*nazimuth_fft*2*4*8;

if(ifocus==2 || ifocus==3)
{
	vvr=dvector(nrange_ffth*nazimuth_fft*2);
	vhr=dvector(nrange_ffth*nazimuth_fft*2);
	hvr=dvector(nrange_ffth*nazimuth_fft*2);
	hhr=dvector(nrange_ffth*nazimuth_fft*2);

	m_total += nrange_ffth*nazimuth_fft*2*4*8;
}
	printf("  Memory allocated= %gMB₩n", m_total/1024./1024.);

	/* read data files */
	for(i=ystart, ii=0; i<=yend; i+=ystep, ii++)
	{
		sprintf(fname, "%s/x%dy0.dat", fhead, i);

		if((fp=fopen(fname, "r"))==NULL)
		{
			printf("Error.. can not open %s.₩n", fname);
			exit(0);
		}

		ix=nrange_fft*ii*2;
		for(j=0; j<nrange; j++)
		{
			jx=ix+2*j;
			fscanf(fp, "%lf %lf %lf %lf %lf %lf %lf %lf %lf",
				&freq, vv+jx, vv+jx+1, vh+jx, vh+jx+1,
				hv+jx, hv+jx+1, hh+jx, hh+jx+1);
		}
		fclose(fp);
	}
	printf("  Read data ok₩n");

	/* Range baseband */
	cnshift_x(vv, nazimuth, nrange_fft, nrange/2*(-1));
	cnshift_x(vh, nazimuth, nrange_fft, nrange/2*(-1));
	cnshift_x(hv, nazimuth, nrange_fft, nrange/2*(-1));
	cnshift_x(hh, nazimuth, nrange_fft, nrange/2*(-1));
	printf("  Range baseband ok₩n");

	/* Range compression by inverse FFT */
	cfft1d(vv, nazimuth, nrange_fft, 0, 1);
	cfft1d(vh, nazimuth, nrange_fft, 0, 1);
	cfft1d(hv, nazimuth, nrange_fft, 0, 1);
	cfft1d(hh, nazimuth, nrange_fft, 0, 1);
	printf("  Range compression ok₩n");

	/* Range cut by half */
	for(i=0; i<nazimuth; i++)
	{
		ix=i*nrange_fft*2;
		ix1=i*nrange_ffth*2;
		for(j=0; j<nrange_ffth*2; j++)
		{
			jx=ix+j;
			jx1=ix1+j;
			vv[jx1]=vv[jx];
			vh[jx1]=vh[jx];
			hv[jx1]=hv[jx];
			hh[jx1]=hh[jx];
		}
	}
```

```
	for(i=nrange_ffth*nazimuth*2; i<nrange_fft*nazimuth*2; i++)
	{
		vv[i]=0.;
		vh[i]=0.;
		hv[i]=0.;
		hh[i]=0.;
	}

	/* Output Range Compressed Data */
	strcpy(foutvv, fvv);
	strcpy(foutvh, fvh);
	strcpy(fouthv, fhv);
	strcpy(fouthh, fhh);

	strcat(foutvv, "_0_RangeComp");
	strcat(foutvh, "_0_RangeComp");
	strcat(fouthv, "_0_RangeComp");
	strcat(fouthh, "_0_RangeComp");

	cfout(vv, nazimuth, nrange_ffth, 1, 1, 0, foutvv);
	cfout(vh, nazimuth, nrange_ffth, 1, 1, 0, foutvh);
	cfout(hv, nazimuth, nrange_ffth, 1, 1, 0, fouthv);
	cfout(hh, nazimuth, nrange_ffth, 1, 1, 0, fouthh);

	gen_ers(foutvv, nazimuth, nrange_ffth, delx, delr_fft, 2, "141");
	gen_ers(foutvh, nazimuth, nrange_ffth, delx, delr_fft, 2, "141");
	gen_ers(fouthv, nazimuth, nrange_ffth, delx, delr_fft, 2, "141");
	gen_ers(fouthh, nazimuth, nrange_ffth, delx, delr_fft, 2, "141");


if(ifocus==2 || ifocus==3)
{
	/*****************************************/
	/* Range-Doppler Algorithm for Short Range */
	/*****************************************/

	printf("Range-Doppler Algorithm for Short Range₩n");

	/* Copy data for Range-Doppler Algorithm */
	for(i=0; i<nazimuth_fft*nrange_ffth*2; i++)
	{
		vvr[i]=vv[i];
		vhr[i]=vh[i];
		hvr[i]=hv[i];
		hhr[i]=hh[i];
	}

	/* Azimuth rail baseband */
	cnshift_y(vvr, nazimuth_fft, nrange_ffth, nazimuth/2*(-1));
	cnshift_y(vhr, nazimuth_fft, nrange_ffth, nazimuth/2*(-1));
	cnshift_y(hvr, nazimuth_fft, nrange_ffth, nazimuth/2*(-1));
	cnshift_y(hhr, nazimuth_fft, nrange_ffth, nazimuth/2*(-1));
	printf("  Azimuth Baseband ok₩n");

	/* Azimuth FFT */
	cfft1d(vvr, nazimuth_fft, nrange_ffth, 1, -1);
	cfft1d(vhr, nazimuth_fft, nrange_ffth, 1, -1);
	cfft1d(hvr, nazimuth_fft, nrange_ffth, 1, -1);
	cfft1d(hhr, nazimuth_fft, nrange_ffth, 1, -1);
	printf("  Azimuth FFT ok₩n");

	/* Range Migration and Azimuth Matched Filtering */
	azvv=dvector(nazimuth_fft*2);
	azvh=dvector(nazimuth_fft*2);
	azhv=dvector(nazimuth_fft*2);
	azhh=dvector(nazimuth_fft*2);
	azchirp=dvector(nazimuth_fft*2);
	phase0=2.*phi/lambda;
	for(i=1; i<nrange_ffth; i++)
	{
		R0=i*delr_fft;
		phase1=phase0/R0;

		/* Range Migration and Chirp generation */
		for(j=0; j<nazimuth_fft; j++)
		{
			/* range migration  */
			if(j<nazimuth_fft/2) u=j*delu_fft;
			else u=(j-nazimuth_fft)*delu_fft;
			x=-0.5*lambda*R0*u;

			Rx=sqrt(R0*R0+x*x); /* no approximation at x=xc=0 */
			iRx=(int)(Rx/delr_fft+0.5);
			jx=j*nrange_ffth*2+2*iRx;
			if(iRx<nrange_ffth)
			{
				azvv[2*j]=vvr[jx];
				azvv[2*j+1]=vvr[jx+1];
				azvh[2*j]=vhr[jx];
				azvh[2*j+1]=vhr[jx+1];
```

```
                azhv[2*j]=hvr[jx];
                azhv[2*j+1]=hvr[jx+1];
                azhh[2*j]=hhr[jx];
                azhh[2*j+1]=hhr[jx+1];
        }
        else
        {
                azvv[2*j]=0.;
                azvv[2*j+1]=0.;
                azvh[2*j]=0.;
                azvh[2*j+1]=0.;
                azhv[2*j]=0.;
                azhv[2*j+1]=0.;
                azhh[2*j]=0.;
                azhh[2*j+1]=0.;
        }

        /* azimuth chirp */
        if(j<nazimuth_fft/2) x=j*delx;
        else x=(j-nazimuth_fft)*delx;
        X_rd=R0*lambda/L_az;
        if(fabs(x)<=(X_rd/2.*1.5))
        /* allow 1.5 of nominal beam width */
        {
                phase=phase1*x*x;
                azchirp[2*j]=cos(phase);
                azchirp[2*j+1]=sin(phase);
        }
        else
        {
                azchirp[2*j]=0.;
                azchirp[2*j+1]=0.;
        }

    }
    cfft1d(azchirp, 1, nazimuth_fft, 0, -1);

    /* azimuth matched filtering */
    for(j=0; j<nazimuth_fft; j++)
    {
        jx=j*nrange_ffth*2+2*i;

tmp1=azvv[2*j]*azchirp[2*j]-azvv[2*j+1]*azchirp[2*j+1];

tmp2=azvv[2*j]*azchirp[2*j+1]+azvv[2*j+1]*azchirp[2*j];
                vvr[jx]=tmp1;
                vvr[jx+1]=tmp2;

tmp1=azvh[2*j]*azchirp[2*j]-azvh[2*j+1]*azchirp[2*j+1];

tmp2=azvh[2*j]*azchirp[2*j+1]+azvh[2*j+1]*azchirp[2*j];
                vhr[jx]=tmp1;
                vhr[jx+1]=tmp2;

tmp1=azhv[2*j]*azchirp[2*j]-azhv[2*j+1]*azchirp[2*j+1];

tmp2=azhv[2*j]*azchirp[2*j+1]+azhv[2*j+1]*azchirp[2*j];
                hvr[jx]=tmp1;
                hvr[jx+1]=tmp2;

tmp1=azhh[2*j]*azchirp[2*j]-azhh[2*j+1]*azchirp[2*j+1];

tmp2=azhh[2*j]*azchirp[2*j+1]+azhh[2*j+1]*azchirp[2*j];
                hhr[jx]=tmp1;
                hhr[jx+1]=tmp2;
            }
    }/* end of range migration and matched filtering */
    free(azhh);
    free(azhv);
    free(azvh);
    free(azvv);
    free(azchirp);
    printf("  Range Migration and Matched Filtering ok₩n");

    /* Azimuth Inverse FFT */
    cfft1d(vvr, nazimuth_fft, nrange_ffth, 1, 1);
    cfft1d(vhr, nazimuth_fft, nrange_ffth, 1, 1);
    cfft1d(hvr, nazimuth_fft, nrange_ffth, 1, 1);
    cfft1d(hhr, nazimuth_fft, nrange_ffth, 1, 1);
    printf("  Azimuth Inverse FFT ok₩n");

    /* Azimuth shift by half */
    cnshift_y(vvr, nazimuth_fft, nrange_ffth, nazimuth_fft/2);
    cnshift_y(vhr, nazimuth_fft, nrange_ffth, nazimuth_fft/2);
    cnshift_y(hvr, nazimuth_fft, nrange_ffth, nazimuth_fft/2);
    cnshift_y(hhr, nazimuth_fft, nrange_ffth, nazimuth_fft/2);
    printf("  Azimuth shift by half ok₩n");

    /* Output Range-Doppler SAR Data */
    strcpy(foutvv, fvv);
```

```
		strcpy(foutvh, fvh);
		strcpy(fouthv, fhv);
		strcpy(fouthh, fhh);

		strcat(foutvv, "_1_RD");
		strcat(foutvh, "_1_RD");
		strcat(fouthv, "_1_RD");
		strcat(fouthh, "_1_RD");

		cfout(vvr, nazimuth_fft, nrange_ffth, 1, 1, 0, foutvv);
		cfout(vhr, nazimuth_fft, nrange_ffth, 1, 1, 0, foutvh);
		cfout(hvr, nazimuth_fft, nrange_ffth, 1, 1, 0, fouthv);
		cfout(hhr, nazimuth_fft, nrange_ffth, 1, 1, 0, fouthh);

		gen_ers(foutvv, nazimuth_fft, nrange_ffth, delx, delr_fft, 2, "141");
		gen_ers(foutvh, nazimuth_fft, nrange_ffth, delx, delr_fft, 2, "141");
		gen_ers(fouthv, nazimuth_fft, nrange_ffth, delx, delr_fft, 2, "141");
		gen_ers(fouthh, nazimuth_fft, nrange_ffth, delx, delr_fft, 2, "141");


} /* end of Range-Doppler */

if(ifocus==1 || ifocus==3)
{
		/*****************************************/
		/*   Deramp-FFT Algorithm for Far Range    */
		/*****************************************/

		printf("₩n");
		printf("Deramp-FFT Algorithm for Far Range₩n");

		/* Deramp */
		phase0=2.*phi/lambda;
		for(i=0; i<nazimuth; i++)
		{
			ix=i*nrange_ffth*2;
			x=(i-nazimuth/2)*delx;
			phase1=phase0*x*x;

			for(j=0; j<nrange_ffth; j++)
			{
				jx=ix+2*j;
				R0=j*delr_fft;
				X_hcut=fabs(R0)/R_full*X/2.;
				if(fabs(x)<X_hcut)
				{
					phase=phase1/R0;
					cosp=cos(phase);
					sinp=sin(phase);
				}
				else
				{
					cosp=0.;
					sinp=0.;
				}

				tmp1=vv[jx]*cosp-vv[jx+1]*sinp;
				tmp2=vv[jx]*sinp+vv[jx+1]*cosp;
				vv[jx]=tmp1;
				vv[jx+1]=tmp2;

				tmp1=vh[jx]*cosp-vh[jx+1]*sinp;
				tmp2=vh[jx]*sinp+vh[jx+1]*cosp;
				vh[jx]=tmp1;
				vh[jx+1]=tmp2;

				tmp1=hv[jx]*cosp-hv[jx+1]*sinp;
				tmp2=hv[jx]*sinp+hv[jx+1]*cosp;
				hv[jx]=tmp1;
				hv[jx+1]=tmp2;

				tmp1=hh[jx]*cosp-hh[jx+1]*sinp;
				tmp2=hh[jx]*sinp+hh[jx+1]*cosp;
				hh[jx]=tmp1;
				hh[jx+1]=tmp2;
			}

}
printf("  Deramp ok₩n");

/* Azimuth rail baseband */
cnshift_y(vv, nazimuth_fft, nrange_ffth, nazimuth/2*(-1));
cnshift_y(vh, nazimuth_fft, nrange_ffth, nazimuth/2*(-1));
cnshift_y(hv, nazimuth_fft, nrange_ffth, nazimuth/2*(-1));
cnshift_y(hh, nazimuth_fft, nrange_ffth, nazimuth/2*(-1));
printf("  Azimuth Baseband ok₩n");

/* Azimuth FFT */
cfft1d(vv, nazimuth_fft, nrange_ffth, 1, -1);
cfft1d(vh, nazimuth_fft, nrange_ffth, 1, -1);
```

```
cfft1d(hv, nazimuth_fft, nrange_ffth, 1, -1);
cfft1d(hh, nazimuth_fft, nrange_ffth, 1, -1);
printf("  Azimuth FFT ok₩n");

/* Azimuth shift by half */
cnshift_y(vv, nazimuth_fft, nrange_ffth, nazimuth_fft/2);
cnshift_y(vh, nazimuth_fft, nrange_ffth, nazimuth_fft/2);
cnshift_y(hv, nazimuth_fft, nrange_ffth, nazimuth_fft/2);
cnshift_y(hh, nazimuth_fft, nrange_ffth, nazimuth_fft/2);
printf("  Azimuth shift by half ok₩n");

/* Output Deramp-FFT SAR Data */
strcpy(foutvv, fvv);
strcpy(foutvh, fvh);
strcpy(fouthv, fhv);
strcpy(fouthh, fhh);

strcat(foutvv, "_2_DF");
strcat(foutvh, "_2_DF");
strcat(fouthv, "_2_DF");
strcat(fouthh, "_2_DF");

cfout(vv, nazimuth_fft, nrange_ffth, 1, 1, 0, foutvv);
cfout(vh, nazimuth_fft, nrange_ffth, 1, 1, 0, foutvh);
cfout(hv, nazimuth_fft, nrange_ffth, 1, 1, 0, fouthv);
cfout(hh, nazimuth_fft, nrange_ffth, 1, 1, 0, fouthh);

gen_ers(foutvv, nazimuth_fft, nrange_ffth, delx_fft, delr_fft, 2, "141");
gen_ers(foutvh, nazimuth_fft, nrange_ffth, delx_fft, delr_fft, 2, "141");
gen_ers(fouthv, nazimuth_fft, nrange_ffth, delx_fft, delr_fft, 2, "141");
gen_ers(fouthh, nazimuth_fft, nrange_ffth, delx_fft, delr_fft, 2, "141");

/* Geocoding by Azimuth Resampling */
for(i=0; i<nrange_ffth; i++)
{
        azvv=dvector(nazimuth_fft*2);
        azvh=dvector(nazimuth_fft*2);
        azhv=dvector(nazimuth_fft*2);
        azhh=dvector(nazimuth_fft*2);

        R0=i*delr_fft;
        /*if(R0<R_full) delx_fftr=L_az/2.;
        else delx_fftr=lambda/2./X*R0;
        */
        delx_fftr=lambda/2.*delu_fft*R0;
        for(j=0; j<nazimuth_fft; j++)
        {
                tmp1=(j-nazimuth_fft/2)*delx_fft;
                jx=(int)(tmp1/delx_fftr+nazimuth_fft/2.+0.5);
                if(jx>=0 && jx<nazimuth_fft)
                {
                        ix=j*2;
                        jx1=jx*nrange_ffth*2+2*i;

                        azvv[ix]=vv[jx1];
                        azvv[ix+1]=vv[jx1+1];
                        azvh[ix]=vh[jx1];
                        azvh[ix+1]=vh[jx1+1];
                        azhv[ix]=hv[jx1];
                        azhv[ix+1]=hv[jx1+1];
                        azhh[ix]=hh[jx1];
                        azhh[ix+1]=hh[jx1+1];
                }
        }

        for(j=0; j<nazimuth_fft; j++)
        {
                jx=j*nrange_ffth*2+2*i;
                ix=j*2;

                vv[jx]=azvv[ix];
                vv[jx+1]=azvv[ix+1];
                vh[jx]=azvh[ix];
                vh[jx+1]=azvh[ix+1];
                hv[jx]=azhv[ix];
                hv[jx+1]=azhv[ix+1];
                hh[jx]=azhh[ix];
                hh[jx+1]=azhh[ix+1];
        }

        free(azhh);
        free(azhv);
        free(azvh);
        free(azvv);
}
printf("  Azimuth Resampling ok₩n");

/* Output Deramp-FFT Geocoded SAR Data */
strcpy(foutvv, fvv);
```

```
	strcpy(foutvh, fvh);
	strcpy(fouthv, fhv);
	strcpy(fouthh, fhh);

	strcat(foutvv, "_3_DF_Geo");
	strcat(foutvh, "_3_DF_Geo");
	strcat(fouthv, "_3_DF_Geo");
	strcat(fouthh, "_3_DF_Geo");

	cfout(vv, nazimuth_fft, nrange_ffth, 1, 1, 0, foutvv);
	cfout(vh, nazimuth_fft, nrange_ffth, 1, 1, 0, foutvh);
	cfout(hv, nazimuth_fft, nrange_ffth, 1, 1, 0, fouthv);
	cfout(hh, nazimuth_fft, nrange_ffth, 1, 1, 0, fouthh);

	gen_ers(foutvv, nazimuth_fft, nrange_ffth, delx_fft, delr_fft, 2, "141");
	gen_ers(foutvh, nazimuth_fft, nrange_ffth, delx_fft, delr_fft, 2, "141");
	gen_ers(fouthv, nazimuth_fft, nrange_ffth, delx_fft, delr_fft, 2, "141");
	gen_ers(fouthh, nazimuth_fft, nrange_ffth, delx_fft, delr_fft, 2, "141");

} /* end of Deramp-FFT */

	printf("₩n");

	clock_end=time(NULL);
	clock_duration=clock_end - clock_start;
	printf("Program finished successfully in %d sec.₩n", clock_duration);
}

void usage()
{
	printf("₩n    Usage: gbsar parameter_file₩n");
}

void intro()
{
	printf("=====================================₩n");
	printf("=       GB-SAR Processor         =₩n");
	printf("=         Version %s           =₩n", Version);
	printf("=                                =₩n");
	printf("=            by                  =₩n");
	printf("=       Hoonyol Lee              =₩n");
	printf("=      Assistant Professor       =₩n");
	printf("=    Department of Geophysics    =₩n");
	printf("=  Kangwon National University    =₩n");
	printf("=     hoonyol@kangwon.ac.kr      =₩n");
	printf("=====================================₩n");
	printf("₩n");
}
```

< Complex FFT: cfft.c >

```
/*********************************
 1D- and 2D-FFT of 2D Complex Data
 H. Lee, 17/7/01
 modified usgin ma.c, 7 May 2004
 Normalized.

    data: double complex in BIP
    nrow, ncol:

    iflag:  0 --- x directional FFT
              1 --- y directional FFT

    isign:  -1 --- forward FFT
             1 --- inverse FFT
            (depends on the definition of FFT)
*********************************/

/*#include "ma.c"*/
#include "twon.c"
#include "dfour1.c"

void cfft1d(data, nrow, ncol, iflag, isign)
double *data;
int nrow, ncol, iflag, isign;
{
	unsigned long nn;
	int i,j, linen;
	double *tmp;

	linen = ncol*2;

	/* x-directional FFT */
	if(iflag == 0)
	{
		tmp = dvector(ncol*2+1);
		nn = twon(ncol);
	for(i=0; i<nrow; i++){
```

```
                for(j=0; j<ncol; j++){
                        tmp[2*j+1] = data[linen*i + 2*j];
                        tmp[2*j+2] = data[linen*i + 2*j + 1];
                }
                dfour1(tmp, nn, isign);
                for(j=0; j<ncol; j++){
                        data[linen*i+2*j] = tmp[2*j+1];
                        data[linen*i+2*j+1] = tmp[2*j+2];
                }
                }
                /* scaling */
                if(isign==1) for(i=0; i<nrow*ncol*2; i++) data[i] /= ncol;
        }

        /* y-directional FFT */
        if(iflag == 1)
        {
                tmp = dvector(nrow*2+1);
                nn = twon(nrow);
                for(i=0; i< ncol; i++){
                        for(j=0; j<nrow; j++){
                                tmp[2*j+1] = data[linen*j+2*i];
                                tmp[2*j+2] = data[linen*j+2*i+1];
                        }
                        dfour1(tmp, nn, isign);
                        for(j=0; j<nrow; j++){
                                data[linen*j+2*i] = tmp[2*j+1];
                                data[linen*j+2*i+1] = tmp[2*j+2];
                        }
                }
                /* scaling */
                if(isign==1) for(i=0; i<nrow*ncol*2; i++) data[i] /= nrow;
        }

        free(tmp);

}

void cfft2d(data, nrow, ncol, isign)
double *data;
int nrow, ncol, isign;
{
        cfft1d(data, nrow, ncol, 0, isign);
        cfft1d(data, nrow, ncol, 1, isign);
}
```

```
void cfft1d_ri(realdata, imagdata, nrow, ncol, iflag, isign)
double *realdata, *imagdata;
int nrow, ncol, iflag, isign;
{
        int i;
        double *data;

        data=dvector(nrow*ncol*2);

        for(i=0; i<nrow*ncol; i++)
        {
                data[2*i]= realdata[i];
                data[2*i+1]=imagdata[i];
        }

        cfft1d(data, nrow, ncol, iflag, isign);

        for(i=0; i<nrow*ncol; i++)
        {
                realdata[i]=data[2*i];
                imagdata[i]=data[2*i+1];
        }

        free(data);

}

void cfft2d_ri(realdata, imagdata, nrow, ncol, isign)
double *realdata, *imagdata;
int nrow, ncol, isign;
{
        cfft1d_ri(realdata, imagdata, nrow, ncol, 0, isign);
        cfft1d_ri(realdata, imagdata, nrow, ncol, 1, isign);
}
```

< Data output : cfout.c >

```
/*********************************************************
                cfout.c

        Complex File OUTput program
```

```
                    * * *

   cfout(data, nrow, ncol, istyle, iband, ireal, ftail)

          1. data (input data): double, IQ, BIP
          2. nrow: number of row
          3. ncol: number of col
          4. istyle (output style): "IQ"(0)  or "amp/phase"(1)
          5. iband (output band format): BIP(0), BIL(1), BSQ(2)
          6. ireal (output data format): float(0) or double(1)
          7. ftail: filename tail (fname=fhead+ftail)

Currently istyle=(0,1), iband=(0,1), ireal=(0,1) are available.

by Dr. Hoonyol Lee
Department of Geophysics
Kangwon National University

* BIL enabled on 28 Nov 2004
* coding initiated on 7 May 2004
*********************************************************/

void cfout(cdata, nrow, ncol, istyle, iband,
ireal, ftail)
double *cdata;
int nrow, ncol, istyle, iband, ireal;
char *ftail;
{

          FILE *fp;
          char fname[80];
          int i, j, ix;
          double *dbuf, x1, y1, amplitude, phase;
          float *fbuf;

          strcpy(fname, ftail);

          printf("    cfout: %s", fname);

          if((fp=fopen(fname, "w"))==NULL)
          {
                    printf(" (Error) Can not open %s.₩n", fname);
                    exit(0);
          }

if(istyle==0) /* "IQ" */
{
   if(iband==0) /* BIP */
   {
          if(ireal==0)
          {
             printf(" (IQ, BIP, FLOAT)₩n");
             fbuf=fvector(ncol*2);
             for(i=0; i<nrow; i++)
             {
                    ix=i*ncol*2;
                    for(j=0; j<ncol*2; j++) fbuf[j]=(float)cdata[ix+j];
                    fwrite(fbuf, 1, ncol*sizeof(float)*2, fp);
             }
             free(fbuf);
          }
          else
          {
             printf(" (IQ, BIP, DOUBLE)₩n");
             fwrite(cdata, 1, nrow*ncol*2*sizeof(double), fp);
          }
   }
   else if(iband==1) /* BIL */
   {
          if(ireal==0)
          {
             printf(" (IQ, BIL, FLOAT)₩n");
             fbuf=fvector(ncol*2);
             for(i=0; i<nrow; i++)
             {
                    ix=i*ncol*2;
                    for(j=0; j<ncol; j++)
                    {
                              fbuf[j]=(float)cdata[ix+2*j];
                              fbuf[ncol+j]=(float)cdata[ix+2*j+1];
                    }
                    fwrite(fbuf, 1, ncol*sizeof(float)*2, fp);
             }
             free(fbuf);
          }
          else
          {
             printf(" (IQ, BIL, DOUBLE)₩n");
             dbuf=dvector(ncol*2);
             for(i=0; i<nrow; i++)
```

```
            {
                ix=i*ncol*2;
                for(j=0; j<ncol; j++)
                {
                    dbuf[j]=cdata[ix+2*j];
                    dbuf[ncol+j]=cdata[ix+2*j+1];
                }
                fwrite(dbuf, 1, ncol*sizeof(double)*2, fp);
            }
            free(dbuf);
        }
    }
    else /* BSQ */
    {
        printf(" (Error) Sorry this format is not ready yet.₩n");
        exit(0);
    }
}
else /* "amplitude, phase" */
{
    if(iband==0) /* BIP */
    {
      if(ireal==0) /* FLOAT */
      {
        printf(" (AP, BIP, FLOAT)₩n");
        fbuf = fvector(ncol*2);
        for(i=0; i<nrow; i++)
        {
            ix=i*ncol*2;
            for(j=0; j<ncol; j++)
            {
                x1=cdata[ix+2*j];
                y1=cdata[ix+2*j+1];

                amplitude=sqrt(x1*x1+y1*y1);
                phase=atan2(y1, x1);

                fbuf[2*j]=(float)amplitude;
                fbuf[2*j+1]=(float)phase;
            }
            fwrite(fbuf, 1, ncol*sizeof(float)*2, fp);
        }
        free(fbuf);
      }
      else /* DOUBLE */
      {
        printf(" (AP, BIP, DOUBLE)₩n");
        dbuf = dvector(ncol*2);
        for(i=0; i<nrow; i++)
        {
            ix=i*ncol*2;
            for(j=0; j<ncol; j++)
            {
                x1=cdata[ix+2*j];
                y1=cdata[ix+2*j+1];

                amplitude=sqrt(x1*x1+y1*y1);
                phase=atan2(y1, x1);

                dbuf[2*j]=amplitude;
                dbuf[2*j+1]=phase;
            }
            fwrite(dbuf, 1, ncol*sizeof(double)*2, fp);
        }
        free(dbuf);
      }
    }
}
else if(iband==1) /* BIL */
{
  if(ireal==0) /* FLOAT */
  {
        printf(" (AP, BIL, FLOAT)₩n");
        fbuf = fvector(ncol*2);
        for(i=0; i<nrow; i++)
        {
            ix=i*ncol*2;
            for(j=0; j<ncol; j++)
            {
                x1=cdata[ix+2*j];
                y1=cdata[ix+2*j+1];

                amplitude=sqrt(x1*x1+y1*y1);
                phase=atan2(y1, x1);

                fbuf[j]=(float)amplitude;
                fbuf[ncol+j]=(float)phase;
            }
            fwrite(fbuf, 1, ncol*sizeof(float)*2, fp);
        }
        free(fbuf);
```

```
            }
            else
            {
                printf(" (AP, BIL, DOUBLE)₩n");
                dbuf = dvector(ncol*2);
                for(i=0; i<nrow; i++)
                {
                    ix=i*ncol*2;
                    for(j=0; j<ncol; j++)
                    {
                        x1=cdata[ix+2*j];
                        y1=cdata[ix+2*j+1];

                        amplitude=sqrt(x1*x1+y1*y1);
                        phase=atan2(y1, x1);

                        dbuf[j]=amplitude;
                        dbuf[j+ncol]=phase;
                    }
                    fwrite(dbuf, 1, ncol*sizeof(double)*2, fp);
                }
                free(dbuf);
            }
        }
        else /* BSQ */
        {
            printf(" (Error) Sorry BSQ format is not supported yet.₩n");
            exit(0);
        }
    }

    fclose(fp);

}
```

< Cyclic shift of data: cnshift.c >

```
/****************************************
        cnshift.c

      : cyclic shift of complex double data in BIP
        by integer n

27 Oct 2006: cnshift
24 July 2006: cshift
by Hoonyol Lee
****************************************/
/*#include "ma.c"*/

void cnshift_x(double *data, int nrow, int ncol, int nshift)
{
        int i, j, ix, jj, ncol2, nshift2;
        double *tmp;

        tmp=dvector(ncol*2);
        ncol2=ncol*2;
        nshift2=nshift*2;

        for(i=0; i<nrow; i++)
        {
                ix=i*ncol2;
                for(j=0; j<ncol2; j++)
                {
                        jj=j+nshift2;
                        while(jj>=ncol2) jj-=ncol2;
                        while(jj<0) jj+=ncol2;
                        tmp[jj]=data[ix+j];
                }
                for(j=0; j<ncol2; j++) data[ix+j]=tmp[j];
        }
        free(tmp);

}

void cnshift_y(double *data, int nrow, int ncol, int nshift)
{
        int i, j, jj, ncol2;
        double *tmp;

        tmp=dvector(nrow);
        ncol2=ncol*2;

        for(i=0; i<ncol2; i++)
        {
                for(j=0; j<nrow; j++)
                {
                        jj=j+nshift;
                        while(jj>=nrow) jj-=nrow;
```

```
                        while(jj<0) jj+=nrow;
                        tmp[jj]=data[j*ncol2+i];
                }
                for(j=0; j<nrow; j++) data[j*ncol2+i]=tmp[j];
        }
        free(tmp);
}

void cnshift_xy(double *data, int nrow, int ncol, int xshift, int yshift)
{
        cnshift_x(data, nrow, ncol, xshift);
        cnshift_y(data, nrow, ncol, yshift);
}
```

< 1D FFT: dfour1.c >

```
 /*#include <math.h>*/
#define SWAP(a,b) tempr=(a);(a)=(b);(b)=tempr

void dfour1(double data[], unsigned long nn, int isign)
{
        unsigned long n,mmax,m,j,istep,i;
        double wtemp,wr,wpr,wpi,wi,theta;
        double tempr,tempi;

        n=nn << 1;
        j=1;
        for (i=1;i<n;i+=2) {
                if (j > i) {
                        SWAP(data[j],data[i]);
                        SWAP(data[j+1],data[i+1]);
                }
                m=n >> 1;
                while (m >= 2 && j > m) {
                        j -= m;
                        m >>= 1;
                }
                j += m;
        }
        mmax=2;
        while (n > mmax) {
                istep=mmax << 1;
                theta=isign*(6.28318530717959/mmax);
                wtemp=sin(0.5*theta);
                wpr = -2.0*wtemp*wtemp;
                wpi=sin(theta);
                wr=1.0;
                wi=0.0;
                for (m=1;m<mmax;m+=2) {
                        for (i=m;i<=n;i+=istep) {
                                j=i+mmax;
                                tempr=wr*data[j]-wi*data[j+1];
                                tempi=wr*data[j+1]+wi*data[j];
                                data[j]=data[i]-tempr;
                                data[j+1]=data[i+1]-tempi;
                                data[i] += tempr;
                                data[i+1] += tempi;
                        }
                        wr=(wtemp=wr)*wpr-wi*wpi+wr;
                        wi=wi*wpr+wtemp*wpi+wi;
                }
                mmax=istep;
        }
}
#undef SWAP
/* (C) Copr. 1986-92 Numerical Recipes Software  */
```

< Output ERMapper Header: gen_ers.c >

```
void gen_ers (fname, nrow, ncol, drow, dcol, nband, bflag)
char fname[], bflag[];
int  ncol, nrow, nband;
double drow, dcol;
{
        char fnamers[80];
        FILE *fp;
        int i, nbyte;

        strcpy(fnamers, fname);
        strcat(fnamers, ".ers");
        if((fp = fopen(fnamers, "w"))==NULL)
        {
                printf("Error... Can not open %s.₩n", fnamers);
                exit(0);
        }
```

```
        fprintf(fp,"DatasetHeader Begin₩n");
        fprintf(fp,"        Version         = ₩"6.0₩"₩n");
        fprintf(fp,"        Name          = ₩"%s₩"₩n", fnamers);
        fprintf(fp,"        Description                 = ₩"Generated by Hoonyol
Lee₩"₩n");
        fprintf(fp,"        LastUpdated     = Thu Jun 12 00:00:00 GMT 2004₩n");
        fprintf(fp,"        SensorName       = ₩"Unknown₩"₩n");
        fprintf(fp,"        SenseDate       = Thu Jun 12 00:00:00 GMT 2004₩n");
        fprintf(fp,"        DataSetType     = ERStorage₩n");
        fprintf(fp,"        DataType        = Raster₩n");
        fprintf(fp,"        ByteOrder       = LSBFirst₩n");
        fprintf(fp,"        CoordinateSpace Begin₩n");
        fprintf(fp,"                Datum           = ₩"RAW₩"₩n");
        fprintf(fp,"                Projection      = ₩"RAW₩"₩n");
        fprintf(fp,"                Projection      = ₩"RAW₩"₩n");
        fprintf(fp,"                CoordinateType  = RAW₩n");
        fprintf(fp,"                Rotation        = 0:0:0.0₩n");
        fprintf(fp,"        CoordinateSpace End₩n");
        fprintf(fp,"        RasterInfo Begin₩n");

        if(strncmp(bflag, "010", 3) == 0)
                fprintf(fp,"                                CellType                =
Unsigned8BitInteger₩n");
        else if(strncmp(bflag, "110",3) == 0)
                fprintf(fp,"                CellType        = Signed8BitInteger₩n");
        else if(strncmp(bflag, "020",3) == 0)
                fprintf(fp,"                                CellType                =
Unsigned16BitInteger₩n");
        else if(strncmp(bflag, "120",3) == 0)
                fprintf(fp,"                    CellType        = Signed16BitInteger
₩n");
        else if(strncmp(bflag, "040",3) == 0)
                fprintf(fp,"                                CellType                =
Unsigned32BitInteger₩n");
        else if(strncmp(bflag, "140",3) == 0)
                fprintf(fp,"                    CellType        = Signed32BitInteger
₩n");
        else if(strncmp(bflag, "141",3) == 0)
                fprintf(fp,"                CellType        = IEEE4ByteReal₩n");
        else if(strncmp(bflag, "181",3) == 0)
                fprintf(fp,"                CellType        = IEEE8ByteReal₩n");
         else
         {          nbyte = atoi(bflag);
                fprintf(fp,"                                CellType                =
Unsigned%dBitInteger₩n", nbyte);
         }

        fprintf(fp,"                CellInfo Begin₩n");
        fprintf(fp,"                        Xdimension      = %lf₩n", dcol);
        fprintf(fp,"                        Ydimension      = %lf₩n", drow);
        fprintf(fp,"                CellInfo End₩n");
        fprintf(fp,"                NrOfLines       = %d₩n", nrow);
        fprintf(fp,"                    RegistrationCellX           = 0₩n");
        fprintf(fp,"                    RegistrationCellY           = %d₩n", nrow/2);
        fprintf(fp,"                NrOfCellsPerLine        = %d₩n", ncol);
        fprintf(fp,"                NrOfBands       = %d₩n", nband);


        for(i=0; i<nband; i++){
                fprintf(fp,"                BandId Begin₩n");
                fprintf(fp,"                                        Value                   =
₩"Band%d₩"₩n", i+1);
                fprintf(fp,"                BandId End₩n");
        }

        fprintf(fp,"        RasterInfo End₩n");
        fprintf(fp,"DatasetHeader End₩n");
        fclose(fp);
}


void ers_usage()
{
        printf(" Usage: gen_ers datafile nrow ncol nband bit_flag(or nbyte)₩n ");
        printf("₩n");
        printf("          where datafile : input file name₩n");
        printf("                  nrow : number of row₩n");
        printf("                  ncol : number of column₩n");
        printf("                 nband : number of band₩n₩n");
        printf("                 nbyte : number of byte per cell₩n₩n");
        printf("              bit_flag : bit flag₩n₩n");
        printf("                         number of byte, or ₩n");
        printf("                         010    Unsigned  8 bit Integer₩n");
        printf("                         110      Signed  8 bit Integer₩n");
        printf("                         020    Unsigned 16 bit Integer₩n");
        printf("                         120      Signed 16 bit Integer₩n");
        printf("                         040    Unsigned 32 bit Integer₩n");
        printf("                         140      Signed 32 bit Integer₩n");
        printf("                         141        IEEE  4 Byte Real₩n");
        printf("                         181        IEEE  8 Byte Real₩n");
```

```
        printf("₩n");
}
```

< Memory allocation: ma.c >

```
/* memory allocation for vectors and matrices */
/* modified from nrutil.c of Numerical Recipes in C */
/* index notation follows C convention */

/*#include <stdlib.h>*/

unsigned char *ucvector(int n)
{
unsigned char *v;

v = (unsigned char *)calloc((size_t) n, (size_t) sizeof( unsigned char));
if(!v)
{       printf("malloc failure in unsigned char vector₩n");
exit(0);
}

return v;
}

char *cvector(int n)
{
char *v;

v = (char *)calloc((size_t) n, (size_t) sizeof(char));
if(!v)
{       printf("malloc failure in char vector₩n");
exit(0);
}

return v;
}

unsigned short *usvector(int n)
{
unsigned short *v;

v = (unsigned short *)calloc((size_t) n, (size_t) sizeof(unsigned short));
if(!v)
{       printf("malloc failure in unsigned short vector₩n");
exit(0);
}
return v;
}

short *svector(int n)
{
short *v;

v = (short *)calloc((size_t) n, (size_t) sizeof(short));
if(!v)
{       printf("malloc failure in short vector₩n");
exit(0);
}
return v;
}

unsigned int *uivector(int n)
{
unsigned int *v;

v = (unsigned int *)calloc((size_t) n, (size_t) sizeof(unsigned int));
if(!v)
{       printf("malloc failure in unsigned int vector₩n");
exit(0);
}
return v;
}

int *ivector(int n)
{
int *v;

v = (int *)calloc((size_t) n, (size_t) sizeof(int));
if(!v)
{       printf("malloc failure in int vector₩n");
exit(0);
}
return v;
}

float *fvector(int n)
{
        float *v;
```

```
        v = (float *)calloc((size_t) n, (size_t) sizeof(float));
        if(!v)
        {       printf("malloc failure in float vector₩n");
                exit(0);
        }
        return v;
}

double *dvector(int n)
{
double *v;

v = (double *)calloc((size_t) n, (size_t) sizeof(double));
if(!v)
{       printf("malloc failure in double vector₩n");
exit(0);
}
return v;
}

int **imatrix(int nrow, int ncol)
{
int i;
int **m;

/* allocate pointers to rows */
m=(int **)calloc((size_t) nrow, (size_t) sizeof(int*));
if(!m)
{       printf("malloc failure 1 in int matrix₩n");
exit(0);
}

/* allocate rows and set pointers to them */
m[0]=(int *)calloc((size_t) nrow*ncol, (size_t)sizeof(int));
if(!m[0])
{       printf("malloc failure 2 in int matrix₩n");
exit(0);
}

for(i=1; i< nrow; i++) m[i] = m[i-1] + ncol;

return m;
}
```

```
float **fmatrix(int nrow, int ncol)
{
int i;
float **m;

/* allocate pointers to rows */
m=(float **)calloc((size_t) nrow, (size_t) sizeof(float*));
if(!m)
{       printf("malloc failure 1 in float matrix₩n");
exit(0);
}

/* allocate rows and set pointers to them */
m[0]=(float *)calloc((size_t) nrow*ncol, (size_t) sizeof(float));
if(!m[0])
{       printf("malloc failure 2 in float matrix₩n");
exit(0);
}

for(i=1; i< nrow; i++) m[i] = m[i-1] + ncol;

return m;
}

double **dmatrix(int nrow, int ncol)
{
        int i;
        double **m;

        /* allocate pointers to rows */
        m=(double **)calloc((size_t) nrow, (size_t) sizeof(double*));
        if(!m)
{       printf("malloc failure 1 in double matrix₩n");
exit(0);
}

        /* allocate rows and set pointers to them */
        m[0]=(double *)calloc((size_t) nrow*ncol, (size_t) sizeof(double));
if(!m[0])
{       printf("malloc failure 2 in double matrix₩n");
exit(0);
}

        for(i=1; i< nrow; i++) m[i] = m[i-1] + ncol;
```

```
        return m;
}
```

< twon.c>

```
/* program to get nn of 2*nn */
/* H.Lee 17/7/01 */

unsigned long twon(int numb)
{
        unsigned long nn = 1;
        int n;

        n = numb - 1;
        do{
                n >>= 1;
                nn <<= 1;
        }while(n != 0);

        return(nn);
}
```

< Parameter file for the indoor experiment : gbsar2.pam >

```
../data     data_directory
room2               output_file_head(fohead)
5 5.6 201           frequency(start,stop)[GHz],number
0.25                Antenna_azimuth[m]
0 4000 50           Rail(start,stop,step)[mm]
2 2                 FFT_fold(interpolation)
0.                  R_center
3                   Focusing_selection

                    1: Deramp-FFT
                    2: Range-Doppler
                    3: Both
```

< Parameter file for the outdoor experiment : test1_FullRD3.pam >

```
../test1    data_directory
t1_FullRD3          output_header(fohead)
5 5.6 1601          frequency(start,stop)[GHz],number
0.25                Antenna_azimuth[m]
0 5000 50           Rail(start,stop,step)[mm]
2 16                FFT_fold_Range_Azimuth(interpolation)
0                   R_center
3                   Focusing_selection

                    1: Deramp-FFT
                    2: Range-Doppler
                    3: Both
```

< Program log file for indoor experiment: gbsar2.log >

```
====================================
=          GB-SAR Processor        =
=             Version 1.1          =
=                                  =
=                by                =
=           Hoonyol Lee            =
=         Assistant Professor      =
=      Department of Geophysics    =
=  Kangwon National University      =
=       hoonyol@kangwon.ac.kr      =
====================================

<< Range Parameters >>
fstart=5e+09Hz, fend=5.6e+09Hz nrange=201
FFT folds in Rangge and Azimuth = 2 2
Center frequency= 5.3e+09Hz
Wavelength= 0.0565646m
Frequency bandwidth (bandw_r)= 6e+08Hz
Time resolution (delt)= 1.66667e-09s
Range resolution (delr)= 0.249827m
Frequency resolution (delf)= 3e+06Hz
Time maximum(t_max)= 1.66667e-07s
Range maximum(R_max)= 24.9827m
```

```
Frequency iFFT pixel size (nrange_fft)= 512
Time sampling resolution (delt_fft)= 6.52316e-10s
Range sampling pixel size (delr_fft)= 0.0977797m

<< Azimuth parameters >>
L_az = 0.25m. The ystep should be less than 125mm.
ystart= 0mm, yend= 4000mm, ystep= 50mm
Rail scan length(X)= 4m
Azimuth wavenumber resolution(delu)= 0.25[1/m]
Full focusing Range limit (R_full)=17.6789m
Number of azimuth samples (nazimuth)= 81
Azimuth sampling step (dela)= 0.05m
Azimuth FFT size (nazimuth_fft)= 256
Rail scan length for FFT (X_fft)= 12.75m
Azimuth wavenumber pixel size(delu_fft)= 0.0784314[1/m]
Azimuth pixel size =0.0554172m at R=24.9827m.

Deramp-FFT and Range-Doppler Algorithms

Memory allocated= 12MB
Read data ok
Range baseband ok
Range compression ok
cfout: room2_VV_0_RangeComp (AP, BIL, FLOAT)
cfout: room2_VH_0_RangeComp (AP, BIL, FLOAT)
cfout: room2_HV_0_RangeComp (AP, BIL, FLOAT)
cfout: room2_HH_0_RangeComp (AP, BIL, FLOAT)
Range-Doppler Algorithm for Short Range
Azimuth Baseband ok
Azimuth FFT ok
Range Migration and Matched Filtering ok
Azimuth Inverse FFT ok
Azimuth shift by half ok
cfout: room2_VV_1_RD (AP, BIL, FLOAT)
cfout: room2_VH_1_RD (AP, BIL, FLOAT)
cfout: room2_HV_1_RD (AP, BIL, FLOAT)
cfout: room2_HH_1_RD (AP, BIL, FLOAT)
Deramp-FFT Algorithm for Far Range
Deramp ok
Azimuth Baseband ok
Azimuth FFT ok
Azimuth shift by half ok
cfout: room2_VV_2_DF (AP, BIL, FLOAT)
cfout: room2_VH_2_DF (AP, BIL, FLOAT)
cfout: room2_HV_2_DF (AP, BIL, FLOAT)
cfout: room2_HH_2_DF (AP, BIL, FLOAT)
Azimuth Resampling ok
cfout: room2_VV_3_DF_Geo (AP, BIL, FLOAT)
cfout: room2_VH_3_DF_Geo (AP, BIL, FLOAT)
cfout: room2_HV_3_DF_Geo (AP, BIL, FLOAT)
cfout: room2_HH_3_DF_Geo (AP, BIL, FLOAT)

Program finished successfully in 2 sec.
```

< Program log file for outdoor experiment: test1_FullRD3.log >

```
==================================
=          GB-SAR Processor      =
=             Version 1.1        =
=                                =
=                by              =
=           Hoonyol Lee          =
=         Assistant Professor    =
=      Department of Geophysics  =
=  Kangwon National University    =
=       hoonyol@kangwon.ac.kr    =
==================================

<< Range Parameters >>
fstart=5e+09Hz, fend=5.6e+09Hz nrange=1601
FFT folds in Rangge and Azimuth = 2 16
Center frequency= 5.3e+09Hz
Wavelength= 0.0565646m
Frequency bandwidth (bandw_r)= 6e+08Hz
```

Time resolution (delt)= 1.66667e-09s
Range resolution (delr)= 0.249827m
Frequency resolution (delf)= 375000Hz
Time maximum(t_max)= 1.33333e-06s
Range maximum(R_max)= 199.862m
Frequency iFFT pixel size (nrange_fft)= 4096
Time sampling resolution (delt_fft)= 6.51201e-10s
Range sampling pixel size (delr_fft)= 0.0976125m

<< Azimuth parameters >>
L_az = 0.25m. The ystep should be less than 125mm.
ystart= 0mm, yend= 5000mm, ystep= 50mm
Rail scan length(X)= 5m
Azimuth wavenumber resolution(delu)= 0.2[1/m]
Full focusing Range limit (R_full)=22.0986m
Number of azimuth samples (nazimuth)= 101
Azimuth sampling step (dela)= 0.05m
Azimuth FFT size (nazimuth_fft)= 2048
Rail scan length for FFT (X_fft)= 102.35m
Azimuth wavenumber pixel size(delu_fft)= 0.0097704[1/m]
Azimuth pixel size =0.0552276m at R=199.862m.

Deramp-FFT and Range-Doppler Algorithms

Memory allocated= 768MB
Read data ok
Range baseband ok
Range compression ok
cfout: t1_FullRD3_VV_0_RangeComp (AP, BIL, FLOAT)
cfout: t1_FullRD3_VH_0_RangeComp (AP, BIL, FLOAT)
cfout: t1_FullRD3_HV_0_RangeComp (AP, BIL, FLOAT)
cfout: t1_FullRD3_HH_0_RangeComp (AP, BIL, FLOAT)
Range-Doppler Algorithm for Short Range
Azimuth Baseband ok
Azimuth FFT ok
Range Migration and Matched Filtering ok
Azimuth Inverse FFT ok
Azimuth shift by half ok
cfout: t1_FullRD3_VV_1_RD (AP, BIL, FLOAT)
cfout: t1_FullRD3_VH_1_RD (AP, BIL, FLOAT)
cfout: t1_FullRD3_HV_1_RD (AP, BIL, FLOAT)
cfout: t1_FullRD3_HH_1_RD (AP, BIL, FLOAT)

Deramp-FFT Algorithm for Far Range
Deramp ok
Azimuth Baseband ok
Azimuth FFT ok
Azimuth shift by half ok
cfout: t1_FullRD3_VV_2_DF (AP, BIL, FLOAT)
cfout: t1_FullRD3_VH_2_DF (AP, BIL, FLOAT)
cfout: t1_FullRD3_HV_2_DF (AP, BIL, FLOAT)
cfout: t1_FullRD3_HH_2_DF (AP, BIL, FLOAT)
Azimuth Resampling ok
cfout: t1_FullRD3_VV_3_DF_Geo (AP, BIL, FLOAT)
cfout: t1_FullRD3_VH_3_DF_Geo (AP, BIL, FLOAT)
cfout: t1_FullRD3_HV_3_DF_Geo (AP, BIL, FLOAT)
cfout: t1_FullRD3_HH_3_DF_Geo (AP, BIL, FLOAT)

Program finished successfully in 130 sec.

주 의

1. 이 보고서는 한국지질자원연구원에서 과학기술부 기본연구사업으로 수행하는 과제의 위탁연구보고서입니다.
2. 이 보고서 내용을 발표하는 때에는 반드시 한국지질자원연구원에서 수행한 기본연구사업의 위탁연구결과임을 밝혀야 합니다.
3. 국가과학기술 기밀유지에 필요한 내용은 대외적으로 발표 또는 공개하여서는 아니됩니다.