THE JOURNAL OF KOREAN INSTITUTE OF ELECTROMAGNETIC ENGINEERING AND SCIENCE. 2019 Sep.; 30(9), 770～779.

http://dx.doi.org/10.5515/KJKIEES.2019.30.9.770
ISSN 1226-3133 (Print) · ISSN 2288-226X (Online)

병렬 처리를 적용한 PFA의 성능 및 연산 시간 분석

Analysis of Performance and Operation Time in Polar Format Algorithm using Parallel Processing

이 동 주 · 임 상 호*

Dongjoo Lee · Sangho Lim*

요 약

SAR(Synthetic Aperture Radar)는 태양 고도 및 기상 조건에 상관없이 고해상도의 영상을 얻을 수 있는 장점을 가진 레이다이다. 고해상도 영상을 얻는 SAR의 대표적인 알고리즘으로 PFA(Polar Format Algorithm), RMA(Range Migration Algorithm), BPA(Back Projection Algorithm) 등이 있다. 본 논문에서는 점표적 신호를 모의하고, SAR 알고리즘 중에서 polar 형태로 수신된 데이터를 rectangular 형태로 신호 처리를 수행하는 PFA 알고리즘을 사용하였다. CPU affinity 설정으로 각 코어마다 프로세스를 직접 할당하고, 제어할 수 있도록 병렬 처리를 적용한 PFA를 구현하여 성능을 분석하고, 연산 시간을 비교 및 분석하였다.

Abstract

Synthetic aperture radar(SAR) is a radar system that can acquire high-resolution images of targets, regardless of weather and position of the sun. Some representative algorithms used for processing SAR signals and forming high-resolution SAR images from them are polar format algorithm(PFA), range migration algorithm(RMA), and back projection algorithm(BPA). This paper deals with the generation of simulated SAR signals of point targets and their subsequent processing using the PFA, which is a signal-processing algorithm that converts received signals in polar format to rectangular format. In addition, this paper provides an analysis of the performance and operation time of PFA through simulations and by applying parallel processing where each core is controlled by CPU affinity.

Key words: SAR, PFA, Parallel Processing, Operation Time, Image Processing

Ⅰ. 서 론

SAR(Synthetic Aperture Radar)는 항공기나 인공위성에 탑재해야 하는 공간적 제한성으로 안테나 직경을 크게 하지 않으면서도 높은 방위 해상도를 얻을 수 있도록 개발한 방식이다. 탑재체의 움직임으로 얻어지는 안테나 위치의 연속적 변화를 이용하여 개구면이 작은 각 안테나의 위치에서 수신된 연속적인 여러 개의 레이다 수신 신호들을 합성하여 개구면이 큰 안테나로 합성하기에 합성개구(synthetic aperture) 레이다라고 한다.

SAR를 이용하면 태양 고도 및 기상 조건에 상관없이 고해상도의 영상을 얻을 수 있는 장점이 있기에 SAR의

「이 연구는 국방과학연구소의 “한국형 전투기 KF-X AESA 레이다 개발” 연구비의 지원으로 연구되었음.」

한화시스템(Hanwha Systems)

*국방과학연구소(Agency For Defense Development)

· Manuscript received July 12, 2019 ; Revised August 15, 2019 ; Accepted September 19, 2019. (ID No. 20190712-065)

· Corresponding Author: Dongjoo Lee (e-mail: dongjoo.lee@hanwha.com)

역할이 계속 증가하고 있고, 국내외에서도 지속적인 연구가 활발히 진행되고 있다[1].

고해상도 영상을 형성하는 SAR의 대표적인 알고리즘으로 PFA(Polar Format Algorithm), RMA(Range Migration Algorithm), BPA(Back Projection Algorithm) 등이 있다.

PFA는 영상 중심으로부터 같은 거리에서 polar 형태로 수신되는 신호를 거리와 방위 방향으로의 보간(interpolation)을 수행하여 rectangular 형태로 바꾸어 주고, 2D IFFT를 통해 고해상도 영상을 형성하는 알고리즘이다. 수신된 신호를 리샘플링(resampling)하기 때문에 다른 알고리즘에 비해 계산 효율이 좋지만, 근사화에 따른 파면 곡률(wavefront curvature)이 나타나고, 이를 보상하기 위한 기하 보정 단계를 고려해야 한다[2],[3].

RMA는 주파수 신호를 방위방향으로 샘플링 수를 크게 하여 수집하고, 영상 중심 라인을 기준으로 요동을 보상한다. 영상 중심 라인에서 거리가 먼 표적에 대한 보상을 위해 Stolt 보간을 수행하고, 2D IFFT를 통해 최종 영상을 형성한다. RMA는 영상 중심 라인을 기준으로 요동을 보상하기 때문에 PFA에 비해 파면 곡률의 영향을 덜 받지만, 대부분의 단계에서 많은 수의 FFT 연산을 필요로 하기에 PFA보다는 연산량이 많고, 연산 시간이 더 소요된다[4].

BPA는 이상적으로는 요동에 의한 성능 저하가 없는 시간 영역에서 고해상도 영상을 형성하는 알고리즘이다. SAR 영상 획득을 위해서는 이상적인 직선 비행을 해야 하지만, 난기류 및 바람의 영향으로 항공기는 요동을 하게 되고, 요동을 보상하지 않을 경우 성능 저하가 발생하게 된다. 이에 BPA의 경우, 연산량이 매우 큰 단점이 있으나, 큰 요동이 존재하는 환경에서 해상도 및 부엽 성능 측면에서 정확한 영상 형성이 가능하다고 알려져 있다[5].

고해상도 영상 형성 후 잔여 위상오차를 보상하기 위한 단계에서는 PGA(Phase Correction Algorithm)가 주로 사용되고, 본 논문에서도 PGA를 적용하였다[6].

본 논문에서는 요동에 따른 영향을 보완하기 위하여 요동 보상을 수행하고[7], 연산 시간 및 연산량의 효율성 측면에서 PFA를 선택하여 시뮬레이션을 수행하였다. 먼저 점표적에 대한 모의 신호를 생성하고, 리눅스 기반 PC에 C++로 PFA를 구현하여 고해상도 영상에 대한 성능을 검증하고, 플랫폼의 jittering 영향을 최소화하기 위해 범용 API를 사용하는 대신에 직접 제어가 가능하도록 각 코어별 thread를 생성하고, 병렬 처리를 위한 설계를 수행하여 연산 시간을 도출하고 분석하였다.

Ⅱ. 본 론

C++로 구현할 PFA를 수행하고 검증하기 위해 점표적에 대한 모의 신호를 생성한다. 생성한 모의 신호로 PFA로 얻은 고해상도 영상에 대한 성능을 검증하고, 병렬 처리를 통한 연산 시간을 도출하고 분석한다.

실제 적용한 알고리즘인 모의 신호 생성과 PFA에 대한 상세한 설명은 아래에서 서술하겠다.

2-1 기하 모델

SAR 고해상도 영상을 획득하기 위한 기하 모델은 그림 1과 같다. 고해상도 영상의 해상도 성능을 만족하기 위해 항공기는 합성 개구면 거리(synthetic aperture length: SAL)를 비행하면서 spotlight 모드로 실시간으로 영상 중심 점에 빔을 조향하면서 데이터를 획득한다.

2-2 모의 신호 생성

그림 2는 모의 신호를 생성하기 위한 점표적의 위치를

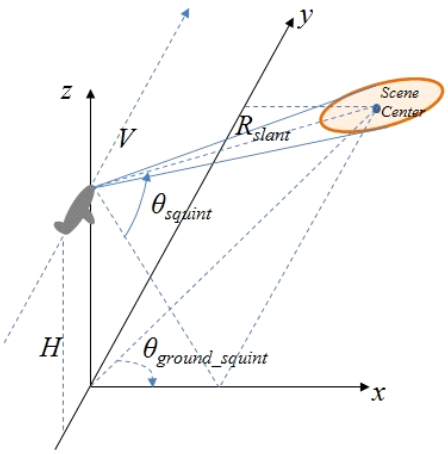

그림 1. SAR의 기하 모델
Fig. 1. Geometry model of SAR.

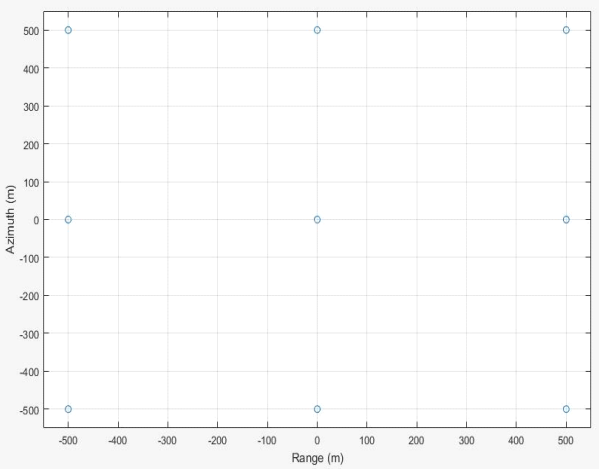

그림 2. 점표적의 위치
Fig. 2. Position of targets.

나타내고 있으며, 기하 모델과 동일한 시나리오로 모의 신호를 생성하였다.

모의 신호 생성을 위해 본 논문에서는 LFM(Linear Frequency Modulation) 펄스를 송신 신호로 사용한다. PFA의 경우, 전파를 송신하고, 수신 시 dechirp을 수행하여 PFA의 입력으로 적용하기에 모의 신호 생성 시에도 dechirp을 수행한 신호를 수신 신호로 사용하게 된다.

PFA에서 사용하는 LFM 수신 신호는 식 (1)과 같고, Reference 신호는 식 (2)와 같이 표현된다. 식 (1)과 식 (2)를 활용하여 dechirp을 수행하면 식 (3)과 같이 표현할 수 있다. 식 (3)에 해당하는 신호를 모의하여 PFA의 입력 신호로 사용하게 된다.

Dechirp이 수행된 식 (3)의 경우, RVP(Residual Video Phase) 성분이 존재하기에 Range deskew 연산을 통해 RVP 성분을 제거하는 단계를 거쳐야 한다. PFA 수행하기 전에 Range deskew 통해 식 (4)와 같이 표현된다.

τ는 거리 방향 시간, η는 방위 방향 시간, a_t는 반사 표적 RCS(Radar Cross Section), w_a는 방위 방향 윈도우, w_r는 거리 방향 윈도우, η_c는 방위 방향의 합성 개구면 시간 SAT(Synthetic Aperture Time), c는 전파의 속도, f_c는 캐리어 주파수, K_r는 chirp rate, $R(\eta)$는 안테나와 표적들 사이의 거리, $R_c(\eta)$는 SAL 중심점에서 표적 지역 중심점과 경사거리를 나타낸다.

$$s_r(\tau,\eta) = a_t w_a(\eta-\eta_c) w_r\left(\tau-\frac{2R(\eta)}{c}\right) \cdot \exp\left(j2\pi f_0\left(\tau-\frac{2R(\eta)}{c}\right)+j\pi K_r\left(\tau-\frac{2R(\eta)}{c}\right)^2\right) \quad (1)$$

$$s_{ref}(\tau,\eta) = \exp\left(j2\pi f_c\left(\tau-\frac{2R_c(\eta)}{c}\right)+j\pi K_r\left(\tau-\frac{2R_c(\eta)}{c}\right)^2\right) \quad (2)$$

$$s_{\text{if}}(\tau,\eta) = a_t w_a(\eta-\eta_c) w_r\left(\tau-\frac{2R(\eta)}{c}\right) \cdot \exp(j\Phi_{\text{if}}(\tau,\eta))$$
$$where$$
$$\Phi_{\text{if}}(\tau,\eta) = -\frac{4\pi K_r}{c}\left(\frac{f_0}{K_r}+\tau-\frac{2R_c(\eta)}{c}\right)\cdot (R(\eta)-R_c(\eta)) + \frac{4\pi K_r}{c^2}(R(\eta)-R_c(\eta))^2 \quad (3)$$

$$s(\tau,\eta) \approx \exp\left[-j\frac{4\pi K_r}{c}\left(\frac{f_0}{K_r}+\tau-\frac{2R_c(\eta)}{c}\right)\cdot (R(\eta)-R_c(\eta)\right] \quad (4)$$

2-3 PFA 알고리즘

PFA로 고해상도 영상을 형성하기 위한 단계 순서도는 그림 3과 같다. Dechrip 수행된 수신 신호에서 RVP를 제거하기 위해 range deskew를 수행하고, 계산의 효율성 및 영상의 patch 영역만 획득하기 위해 거리 방향으로 decimation 과정을 수행한다. 항공기 운행 시 발생하는 요동을 보상하기 위한 요동 보상을 수행하고, 방위 방향도 마찬가지로 영상의 patch 영역만 획득하고 계산하기 위해 decimation을 수행한다. Polar 형태로 수신되는 신호를 보상하기 위해 거리 보간 및 방위 보간을 수행하여 rectangular 형태로 변환한다. 윈도우를 적용하고, 2D IFFT를 수행하여 2D 고해상도 영상을 형성 후 잔여 위상 오차를 보상하기 위해 autofocus를 수행하고, 기하 오차 보정을 통해 최종 영상을 형성한다.

본 논문에서는 점표적에 해당하는 모의 신호를 생성하여 시뮬레이션을 수행하였고, 기하 오차 보정 단계는 수행하지 않고 autofocus까지 수행하여 최종 고해상도 영상을 형성하였다.

PFA에서 3차원 레이다 구조는 그림 4와 같다. s는 점

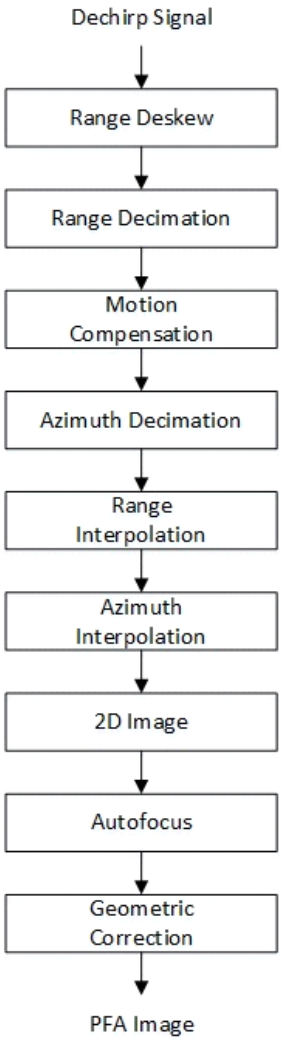

그림 3. Polar format algorithm 순서도
Fig. 3. Flowchart of polar format algorithm.

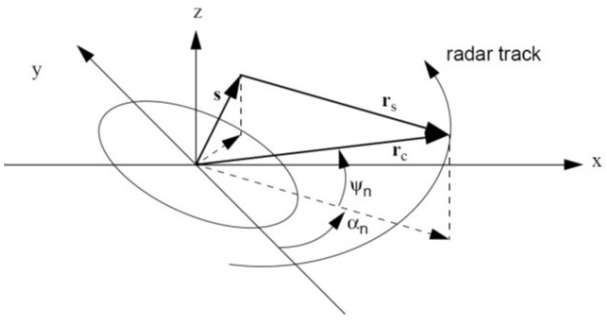

그림 4. 3차원 레이다 구조
Fig. 4. Data collection geometry in a 3D world.

표적의 위치, r_c는 레이다의 위치, r_s는 레이다와 점표적의 상대적 위치이고, 벡터 r_c는 각도 α_n, ψ_n으로 표현된다.

PFA에서 점표적 위치 (s_x, s_y, s_z)에서 수신되는 신호를 dechrip과 analog to digital converter를 수행한 후 데이터를 파수 영역(wavenumber domain)항으로 표현하면 식 (5)와 같다.

$$X_V(i,n) = A_R\sigma(s)\exp j(k_x(i,n)s_x + k_y(i,n)s_y + k_z(i,n)s_z)$$
$$where$$
$$k_x(i,n) = \frac{2}{c}(w_n + \gamma_n T_{s,n} i)\cos\psi_n \sin\alpha_n$$
$$k_y(i,n) = -\frac{2}{c}(w_n + \gamma_n T_{s,n} i)\cos\psi_n \cos\alpha_n$$
$$k_z(i,n) = \frac{2}{c}(w_n + \gamma_n T_{s,n} i)\sin\psi_n \quad (5)$$

w_n는 송신 펄스의 중심 주파수, τ_n는 송신 펄스의 chirp rate, A_R는 송신 펄스의 크기, $T_{s,n}$는 샘플링 시간, c는 전파의 속도, i는 거리 방향의 샘플 번호, n는 펄스 번호이다.

Polar 형태로 수신된 신호를 rectangular 형태로 변환하기 위해 거리 보간 및 방위 보간을 수행한다. $k_y(i,n)$을 중심 펄스 기준($\alpha_n = 0$)으로 펄스 번호 n에 대하여 변하지 않는 값을 가지는 $k_y(i',0)$로 보간한다. 정리하면 식 (6)과 같이 표현된다. 식 6을 풀어서 정리하면 식 (7)~식 (9)와 같이 정리된다.

$$k_y(i,n) = k_y(i',0) = k_y(i') \quad (6)$$

$$(w_n + \gamma_n T_{s,n} i)\cos\psi_n \cos\alpha_n = (w_0 + \gamma_0 T_{s,0} i)\cos\psi_0 \quad (7)$$

$$(w_n + \gamma_n T_{s,n} i) = (w_0 + \gamma_0 T_{s,0} i')\frac{\cos\psi_0}{\cos\psi_n \cos\alpha_n} \quad (8)$$

$$k_y(i') = \frac{2}{c}(w_0 + \gamma_0 T_{s,0} i')\left(\frac{\cos\psi_0}{\cos\psi_n \cos\alpha_n}\right) \quad (9)$$

$k_x(i',n)$ 과 $k_z(i',n)$을 식 (9)를 이용해서 정리하면 식 (10) 및 식 (11)과 같이 표현된다.

$$k_x(i',n) = \frac{2}{c}(w_0 + \gamma_0 T_{s,0} i')\cos\psi_0 \tan\alpha_n \quad (10)$$

$$k_z(i',n) = \frac{2}{c}(w_0 + \gamma_0 T_{s,0} i')\left(\frac{\cos\psi_0}{\cos\psi_n}\right)\tan\psi_n \quad (11)$$

거리 보간 및 방위 보간을 수행 후 파수 영역으로 표현된 식을 윈도우 적용 후 2D IFFT 수행하여 고해상도 영상을 형성한다. 또한 고해상도 영상의 잔여 위상 오차를 보상하기 위하여 autofocus를 N번 수행하여 최종 고해상도 영상을 형성한다.

본 논문에서는 autofocus 단계를 5회 수행하는 것으로 가정하여 최종 고해상도 영상을 형성하고, 연산 시간을 도출하였다.

요동 보상의 경우, 항공기에서 SAR 영상을 형성 시에 외부 환경적인 요인에 의해 요동이 발생하고, 이에 따라 SAT 동안 위상 차이가 발생하여 최종 고해상도 영상의 해상도 및 성능에 많은 영향을 미치게 된다.

항공기 이동에 대한 모의와 그에 따른 IMU(Inertial Measurement Unit) 및 EGI(Embedded GPS Inertial) 값을 모의하여 요동에 대한 값을 측정하고, 모의하여 측정된 요동 측정값을 활용하여 요동 보상을 수행하였다.

Ⅲ. 병렬 처리

3-1 병렬 처리

과거 초기의 CPU는 한 순간에 하나의 일만 수행할 수 있었다. 작업이 끝나면 다 작업을 수행하는 방식이었다. 이 후 단일 프로세스에 병렬 기법이 도입되어 여러 작업을 동시에 처리할 수 있게 되는 병렬 처리 시스템이 나타나게 되었다. 병렬 처리는 컴퓨터를 병렬로 연결하거나, CPU 등을 병렬로 연결하여 다수의 프로세서들이 다수의 프로그램들을 분담하여 동시에 처리하는 방식이다. 하나의 운영체제 안에서 병렬로 연결된 자원, 메모리, CPU 또는 컴퓨터가 작업을 처리하는 방식을 총칭하는 의미로 병렬 처리 시스템이라는 용어로 사용된다. 병렬 처리 시스템은 프로세서를 늘려서 여러 일을 동시에 더 빨리 처리할 수 있게 해주는 시스템 방식이다.

일반적인 병렬 프로그램 도구인 OpenMP는 공유메모리를 이용한 멀티스레딩을 만들고, 다양한 플랫폼을 지원하는 표준화된 API이다. 표준화 및 다양한 플랫폼을 지원하고, 쉽게 구현이 가능한 장점이 있다. 하지만 사용하는 플랫폼의 OS가 자체적으로 멀티 코어를 할당하여 수행하기 때문에 OS 내의 처리하는 우선 순위와 OS 자체적으로 발생하는 jittering 영향을 받는다.

그림 5와 그림 6은 OpenMP 사용에 따른 정상적인 처리 시간과 jittering 현상이 발생했을 경우의 처리 시간을 나타낸 것이다. Jittering 현상에 대한 확인은 CPU의 코어에 합채널, 2개의 차채널, SLB(Side Lobe Blanking) 채널을 OpenMP로 할당하고, 임의의 신호에 대해 펄스 압축을 수행한 결과이다.

정상적인 동작일 경우, 펄스 압축 수행 시간은 577 usec가 걸리지만, jittering이 발생하는 경우, 7449 usec까지 걸리는 경우가 발생한다.

비행기에 탑재되어 활용되는 경우, 마이크로초 단위까지 제어 및 예측이 가능해야 하고, 다른 여러 기능을 동시

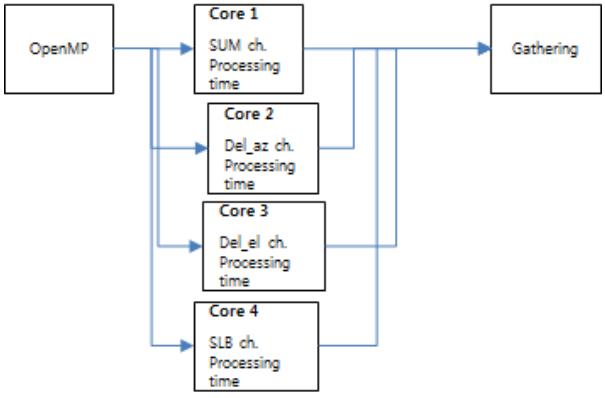

그림 5. OpenMP 사용에 따른 정상적인 동작
Fig. 5. Processing time of OpenMP.

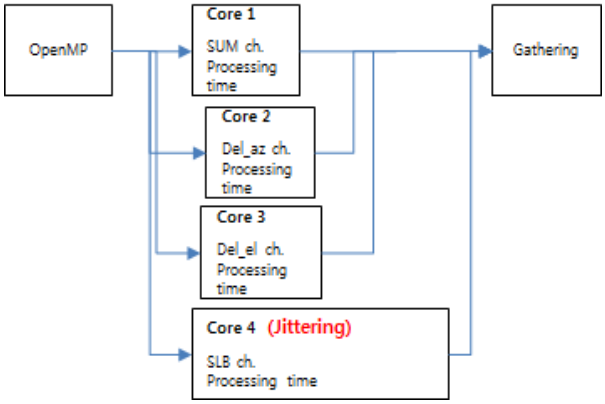

그림 6. OpenMP 사용에 따른 jittering 현상
Fig. 6. Processing time of OpenMP with jittering.

에 동작하게 하려면 직접 코어 할당을 하고, 제어를 수행하는 것이 효율적이다. OpenMP를 사용하여 허용 범위 이상의 jittering이 발생할 경우, 예측하기 어려운 문제에 직면할 수 있다.

이에 본 논문에서는 OpenMP를 사용하는 대신 원하는 프로세스가 특정 CPU 혹은 코어에 할당 및 제한하여 사용할 수 있는 CPU affinity 기능과 OS 운용시 발생할 수 있는 jittering 현상을 최소화할 수 있도록 리눅스 shielding 기능을 적용하여 병렬 처리를 수행하였다. 리눅스 스케쥴러는 적용된 CPU affinity에 따라 프로세스가 지정된 코어 외의 다른 코어에서 돌아가지 않도록 하고, shielding 기능을 통해 인터럽트 등의 외부 환경 요인을 막아준다[8].

PFA의 병렬 처리를 수행하기 위해 각 코어마다 Thread를 할당하여 PFA의 각 단계마다 수행하고자 하는 역할을 해당 Thread, 즉 코어에 할당하여 PFA를 수행하는 연산 시간을 줄이고, 이에 따른 병렬 처리 연산 시간을 측정하였다.

3-2 PFA의 병렬 처리

본 논문에서는 멀티 코어가 적용된 CPU를 사용하여 모의 신호에 대해 PFA의 각 단계마다 CPU 코어를 할당하여 병렬 처리를 수행할 수 있도록 설계하고, 코드 최적화 과정을 거쳐 연산 시간을 줄였다.

시험에 사용한 CPU의 경우, 8개 코어를 사용할 수 있기에 8개 코어를 모두 사용하여 SW 구조 설계 및 구현하여 연산 시간을 도출하였다. 우선 각 단계마다 class로 만들고 Project Manager에 등록하고, 각 CPU 코어에서 실행할 수 있도록 Thread를 생성한다. 수행하고자 하는 코어와 처리하고자하는 데이터 시작점과 끝점을 설정하여 병렬 처리를 수행하도록 설계하였다. 그림 7에 간략하게 도식화하였다.

실제 항공기 적용시 Range deskew는 SAT 동안 처리할 수 있기에 생성한 모의 신호에 대해 SAT 동안 펄스마다 CPU의 각 코어에 할당하여 병렬 처리를 수행하고, SAT 동안 처리된 결과를 순차적으로 메모리에 저장한다.

SAT 이후 수행되는 거리 Decimation, 요동 보상 등의 이후 단계는 메모리에 저장된 Range deskew 결과 데이터

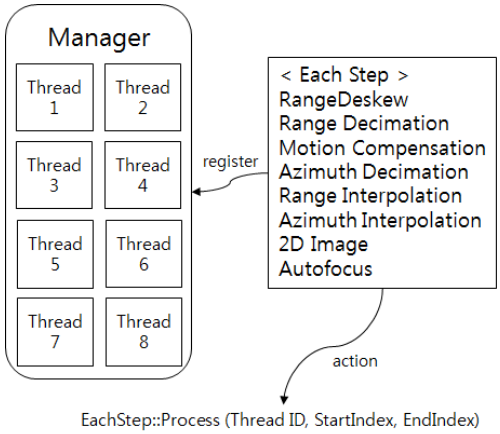

그림 7. 병렬 처리를 위한 SW 설계
Fig. 7. SW architecture for parallel processing.

를 이용한다.

PFA의 각 단계는 거리 방향 혹은 방위 방향으로 독립적인 관계에 있다. 거리 decimation, 요동 보상, 거리 보간의 단계는 각각 거리 방향으로 독립적이기에 거리 방향의 묶음으로 병렬 처리를 수행할 수 있다. 그 외의 방위 decimation, 방위 보간 등은 방위 방향의 묶음으로 병렬 처리를 수행할 수 있다.

2D 고해상도 영상 형성 단계의 경우, 거리 방향으로 윈도우 및 IFFT 수행 후 방위 방향으로 윈도우 및 IFFT 수행하여 최종영상을 생성하였다. 다만, autofocus 단계의 경우 계산 특성상 병렬 처리를 수행하지 않고 2D 고해상도 영상 형성 결과를 1개의 코어에서 처리하였다.

그림 8은 PFA 각 단계별 거리 방향과 방위 방향에 따른 병렬 처리 방법을 표현하였다. 각 코어에서 수행하는 데이터의 시작점과 끝점은 전체 데이터 길이에서 8개 코어에 똑같이 분배되도록 계산하여 적용하였다.

본 논문에서는 멀티 코어가 적용된 CPU를 사용하여 PFA의 각 단계마다 코드 최적화 및 병렬 처리를 적용하여 연산 시간을 줄이도록 SW 설계 및 구현하여 연산 시간을 측정하였다.

생성한 모의 신호를 이용하여 항공기에 적용되는 환경과 동일하게 Range deskew, 거리 Decimation, 요동 보상, 거리 보간 등은 거리 방향의 묶음으로, 방위 Decimation, 방위 보간 등은 방위 방향의 묶음으로 병렬 처리를 수행

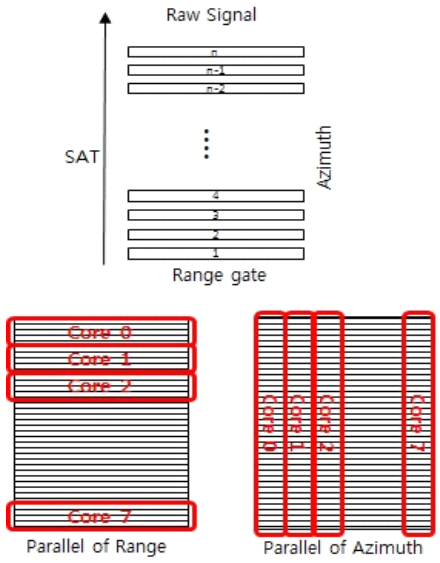

그림 8. PFA의 병렬 처리
Fig. 8. Parallel computing of polar format algorithm.

하였다. 2D 고해상도 영상 형성은 단계에 따라 거리 방향 및 방위 방향으로 병렬 처리를 수행하였고, Autofocus 단계의 경우 계산 특성상 병렬 처리를 수행하지는 않았다.

Ⅳ. 시뮬레이션

4-1 시뮬레이션 환경

그림 2에 해당하는 점표적에 대해 생성한 모의 신호를 이용하여 알고리즘의 성능 및 연산 시간을 리눅스 환경에서 C++로 구현한 시뮬레이션을 통해 확인하였다. 표 1은 사용한 PC 환경을 표기하였고, 표 2는 시뮬레이션을 수행한 파라미터를 표기하였다.

C++ 시뮬레이션 구현에 앞서 기본적인 PFA 알고리즘

표 1. PC 환경
Table 1. Environment of PC.

Environment	Value	
CPU	Intel i7-7700HQ @2.8 GHz	Intel Xeon D @1.5 GHz
Memory	32 GB DDR4	32 GB DDR4
OS	Linux	Linux

표 2. 시뮬레이션 조건
Table 2. Parameter of simulation.

Parameter	Value
Frequency	X-band
Ownship height	8 km
Ownship velocity	200 m/s
Slant range	40 km
Squint angle	40°
Resolution	0.7 m

을 Matlab으로 구현하여 생성한 모의 신호를 이용하여 1차적으로 검증 및 분석을 수행하였다. C++로 구현한 병렬처리가 적용된 PFA 시뮬레이션 결과의 성능 분석은 Matlab에서 제공하는 툴을 사용하여 비교, 검증하였다. 병렬 처리에 따른 분석은 일반적인 처리와의 연산 시간 측면에서 비교하였다.

4-2 성능 분석

그림 2의 점표적을 모의 신호로 생성하여 동일한 위치에 점표적이 표기되는지 여부와 요구하는 해상도를 만족하는지를 확인하였다.

그림 9는 PFA 알고리즘을 통해 얻은 점표적 영상을 표현하였고, 그림 10은 그림 9의 점표적을 확대한 영상이다.

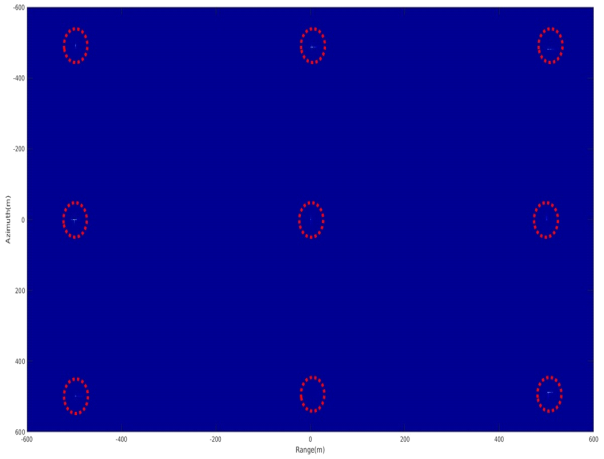

그림 9. PFA 결과 영상
Fig. 9. Image after applying PFA.

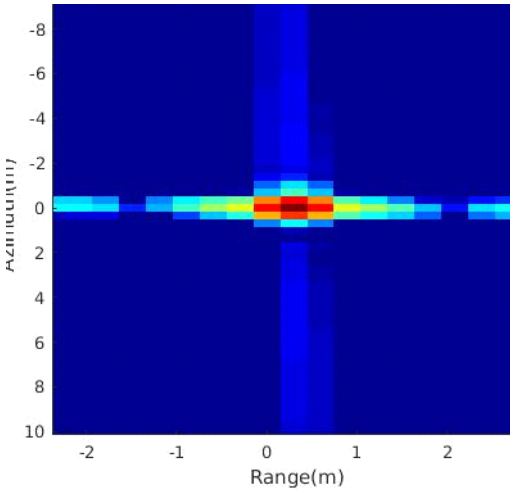

그림 10. 확대한 PFA 결과 영상
Fig. 10. Zoom of PFA image.

그림 11 및 그림 12는 그림 9에서 표현된 점표적들이 설계한 해상도를 만족하는지를 확인한 결과이다.

PFA 알고리즘의 병렬 처리를 통해 얻은 결과 영상의 해상도 측정을 위해 그림 11과 그림 12에서 3 dB 폭에 해당하는 영역을 취하면, 그림 11에서 나타내는 range 방향의 해상도는 0.384 m, 그림 12에서 나타내는 azimuth 방향의 해상도는 0.586 m로 설계한 0.7 m 해상도를 만족함을 알 수 있다. 실제 항공기에 적용시 외부 환경적인 요인으로 발행할 수 있는 오차를 감안한 마진도 반영되었다고

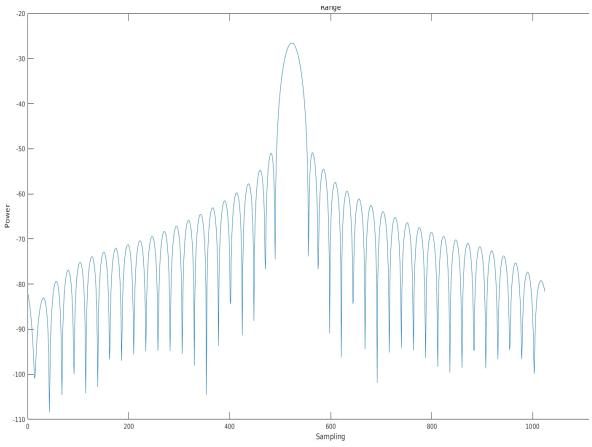

그림 11. 거리 방향의 해상도
Fig. 11. Range resolution of point target.

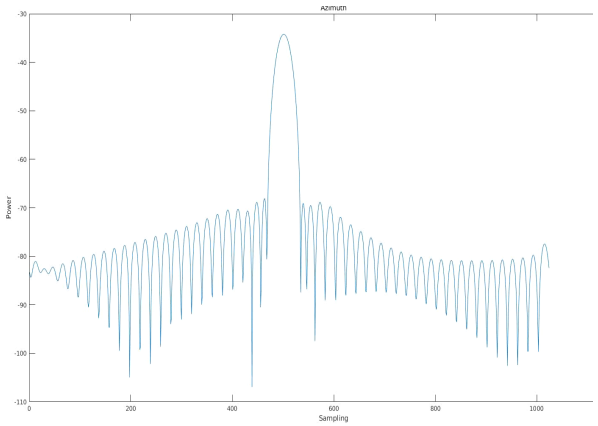

그림 12. 방위 방향의 해상도
Fig. 12. Azimuth resolution of point target.

볼 수 있다.

4-3 연산 시간 분석

PFA 알고리즘의 병렬 처리를 통해 얻은 결과의 성능 분석은 앞 절의 내용을 통해 확인하였고, 이번 절에선 일반적인 방식과 병렬 처리 방식에 따른 PFA 알고리즘 연산 시간을 측정하여 비교하였다.

표 1에서 언급된 PC 환경하에서 표 2의 시나리오를 적용한 PFA 알고리즘 연산 시간을 측정하여 표 3에 표기하였다. 표 3에 표기된 연산 시간은 일반적인 방식과 병렬 처리 방식으로 구현된 코드를 각각 10번 수행한 평균값을 측정한 결과이다.

표 3에 따르면 병렬 처리를 수행하는 단계별로 처리 시간은 PC 환경에 따라 4배 이상 빨리짐을 확인할 수 있다.

Autofocus 단계의 경우, 병렬 처리를 수행하지 않았지만, 병렬 처리를 수행할 수 있도록 알고리즘 수정 및 SW 구조 설계가 보완된다면 추가적인 연산 시간 감소 효과를 볼 수 있을 것이다.

또한 표 1의 2가지 사양의 PC 환경에서 측정하여 CPU의 속도 및 코어 개수에 따른 영향성을 확인해 보고자 다른 사양의 PC 환경에서 병렬 처리 적용된 PFA 알고리즘의 연산 시간을 측정하였다. 이와 별도로 코드 최적화 및 SW 구조 설계에 따라 측정된 연산 시간의 변동이 있을 수 있다.

표 3. 연산 시간 비교
Table 3. Comparison of operation time.

	<PC-1>		<PC-2>	
	Serial processing (ms)	Parallel processing (ms)	Serial processing (ms)	Parallel processing (ms)
Range deskew	6,720.929	1,577.628	1,2136.321	2169.769
Range decimation	2,290.025	532.635	2,1367.035	637.7243
Motion compensation	418.099	63.519	574.248	89.41067
Azimuth decimation	5,406.683	1,254.236	1,2142.713	1,499.734
Range interpolation	711.107	151.811	1,559.658	196.3083
Azimuth interpolation	907.382	243.489	2,207.054	256.3427
2D Image	1,031.398	385.967	4,034.355	567.748
Total	1,7485.62	4,209.288	54,021.384	5,417.037
Autofocus	5609.847		13,740.63	
Total	2,3095.47	9,819.136	67,762.014	19,157.67

Ⅴ. 결 론

본 논문에서는 모의 신호를 생성하고 PFA 알고리즘의 병렬 처리를 수행할 수 있게 구현하여 성능 분석 및 연산 시간을 비교하였다.

스퀸트 각도가 존재하는 시나리오에 따라 최종 영상의 해상도 등을 확인하여 병렬 처리로 구현한 PFA 알고리즘 시뮬레이션 성능을 검증하였고, 병렬 처리를 적용한 PFA 단계에 따른 연산 시간을 도출하고, 일반적인 방식과 비교하였다.

리눅스 기반의 C++로 구현한 PFA 알고리즘 시뮬레이션의 성능 및 연산 시간을 바탕으로 항공기에 적용 가능성을 확인할 수 있었다.

References

[1] A. Moreira, P. Prats-Iraola, M. Younis, G. Krieger, I. Hajnsek, and K. P. Papathanassiou, "A tutorial on synthetic aperture radar," *IEEE Geoscience and Remote Sensing Magazine*, vol. 1, no. 1, pp. 6-43, Mar. 2013.

[2] A. W. Doerry, "Basic of polar-format algorithm for processing synthetic aperture radar images," Sandia National Laboratories, Livermore, SAND2012-3369, May 2012.

[3] A. W. Doerry, "Wavefront curvature limitations and compensation to polar format processing for synthetic aperture radar images," Sandia National Laboratories, Livermore, SAND2007-0046, Jan. 2007.

[4] I. G. Cumming, F. H. Wong, *Digital Processing of Synthetic Aperture Radar Data*, Boston, Artech House, 2005.

[5] A. F. Yegulalp, "Fast backprojection algorithm for synthetic aperture radar," in *Proceedings of the 1999 IEEE Radar Conference*, *Radar into the Next Millennium*, Waltham, MA, 1999, pp. 60-65.

[6] D. E. Wahl, P. H. Eichel, D. C. Ghiglla, and C. V. Jakowatz, "Phase gradient autofocus: A robust tool for high resolution SAR phase correction," *IEEE Trasnsactions on Aerospace and Electronic Systems*, vol. 30, no. 3, pp. 827-835, Jul. 1994.

[7] G. Fornaro, G. Franceschetti, and S. Perna, "Motion compensation errors: Effects on the accuracy of airborne SAR images," *IEEE Aerospace and Electronic Systems Magazine*, vol. 41, no. 4, pp. 1338-1352, Oct. 2005.

[8] S. Brosky, "Shielded CPUs: Real-time performance in standard Linux," *Linux Journal*, May 2004. https://www.linuxjournal.com/article/6900

이 동 주 [한화시스템/전문연구원]

https://orcid.org/0000-0002-0937-7643

2005년 2월: 숭실대학교 정보통신전자공학부 (공학사)
2007년 2월: 연세대학교 전기전자공학부 (공학석사)
2016년 4월: LG전자 CTO 선임연구원
현재: 한화시스템 전문연구원
[주 관심분야] 레이다 신호처리, 영상 처리, 임베디드 SW 등

임 상 호 [국방과학연구소/선임연구원]

https://orcid.org/0000-0001-8250-3784

2006년 2월: 중앙대학교 전자전기공학부 (공학사)
2008년 2월: 한국과학기술원 전기 및 전자공학부 (공학석사)
2011년 8월: 한국과학기술원 전기 및 전자공학과 (공학박사)
2016년 9월: 삼성전자 책임연구원
현재: 국방과학연구소 선임연구원
[주 관심분야] 영상 레이다 및 항공기 레이다의 신호처리 및 시스템 등