JANT 한국항행학회논문지
Journal of Advanced Navigation Technology

항행 및 항법

J. Adv. Navig. Technol. 28(5): 616-622, Oct. 2024

GPU를 이용한 우주감시레이다 신호처리 고속화 방안

High-Speed Signal Processing Using GPU for Space Surveillance Radar

조인철[1] · 조성훈[1] · 안지훈[1] · 문현욱[1*] · 손성환[1] · 정태희[2] · 임상호[2]
[1]LIG넥스원 레이다연구소
[2]국방과학연구소 레이다전자전기술센터

In-cheol Cho[1] · Sung-hoon Cho[1] · Ji-hoon An[1] · Hyun-wook Moon[1*] · Sung-hwan Sohn[1] · Taehee Jeong[2] · Sangho Lim[2]

[1]Radar R&D Lab, LIG Nex1, Gyeonggi-do, 16911, Korea

[2]Radar and EW Technology Center, Agency for Defense Development (ADD), Daejeon, 34186, Korea

[요 약]

최근 들어 각국이 경쟁적으로 우주개발에 막대한 자금을 투자하고 있고 수 만개의 인공위성 및 스파이 위성, 우주 잔해 등 우주 위험이 증가하는 추세다. 이에 따라 우주를 감시하기 위한 여러 자산이 있지만 상시 및 광역 감시가 가능한 레이다를 이용한 우주 물체 탐지에 대한 관심이 높아지고 있다. 기존의 레이다 시스템은 수 백키로 이내의 항공기, 함정, 미사일 등을 탐지하기 위해 운용했다면 우주 물체를 탐지하기 위한 레이다는 수천 키로까지 확장해 운용을 해야한다. 이러한 경우 신호처리를 해야 하는 데이터 크기가 증가하며 이는 레이다 운용에 지연을 초래할 수 있다. 본 논문에서는 레이다를 이용한 우주 물체 탐지시 GPU를 활용해 신호처리를 고속화하는 방법을 제안하고 CPU 대비 성능을 비교해 이를 검증하였다.

[Abstract]

Recently, each country is competitively investing huge amounts of money in space development, and space risks such as tens of thousands of satellites, spy satellites, and space debris are increasing. Although there are several assets for monitoring space, interest is growing in the detection of space objects using radar, which is useful for constant alert and wide-area surveillance. While the existing radar system was operated to detect aircraft, ships, and missiles within hundreds of kilometers, radar to detect space objects must be expanded to thousands of kilometers. In this case, the size of data that impairs signal processing increases, which can lead to a decrease in radar detection performance. In this paper, we present a method of speeding up signal processing using GPU when detecting space objects, which are long-distance targets, and the results of comparing performance compared to existing CPUs.

Key word : CPU, CUDA, GPU, Signal processing, Space surveillance radar.

http://dx.doi.org/10.12673/jant.2024.28.5.616

Received 11 October 2024; **Revised** 27 October 2024
Accepted (Publication) 30 October 2024 (31 October 2024)

***Corresponding Author; Hyun-wook Moon**

Tel: ***-****-****

E-mail: hyunwook.moon2@lignex1.com

Ⅰ. 서 론

1950년대 중반부터 인공위성을 쏘아 올리기 시작해 지금까지 약 6만 개 이상의 인공위성이 다양한 국가에서 운용 중에 있다 [1]. 그만큼 우주 자산을 이용한 정보전 양상이 전 세계적으로 심화되고 있으며, 우주 잔해 낙하로 인한 피해와 스파이 위성 운용으로 자국의 피해가 예상된다 [2]. 이에 따라 레이다 감시 영역이 기존의 영토, 영공, 영해를 넘어 우주로 확장하고 있다. 인공위성과 우주 잔해를 포함한 우주 물체는 기본적으로 저궤도에 위치하는 초장거리 표적이다. 장거리 표적에서 반사된 레이다 신호는 일반적인 단거리 표적의 반사 신호보다 데이터 양이 많아 신호처리 연산시간도 증가하게 되며 이는 레이다 탐지/추적 성능 면에서 성능 저하를 발생시킨다. 이를 해결하기 위해 고사양의 CPU (central processing unit)가 탑재된 DSP (digital signal processor) 사용과 다중 신호처리 HW (hardware) 구성, 병렬처리 등 다양한 방법이 시도되고 있다 [3]. 이 중 고사양 DSP를 사용하는 방법은 처리 시간 효율은 증가하지만 비용 문제가 발생할 수 있으며, 다중 신호처리 HW를 구성하는 방법은 처리해야 할 데이터의 양을 분산시킴으로써 처리 효율을 증가시킬 수 있지만 이 역시 HW 비용은 증가하게 된다. 이러한 문제의 대안으로 저비용으로 수 천개의 코어를 사용할 수 있는 GPU (graphic processing unit)를 활용해 SAR (synthetic aperture radar) 이미지 처리 효율을 증가시키는 병렬처리 연구들이 진행된 바 있다 [4]-[6]. 이러한 GPU를 활용한 병렬처리 방법은 고성능 CPU를 활용한 연산처리 방법에 비해 비교적 적은 비용으로 고성능 연산이 가능한 장점이 있지만 운용 환경 요구사항 충족을 위해 VPX (versatile performance switching)를 사용한 기존 레이다에는 적합하지 않아 적용이 제한되어 왔다. 이에 따라 최근 서버급 HPC (high performance computer) 형태의 처리장치에 GPU를 적용한 연구들이 진행되고 있다 [7], [8]. 그러나 현재까지 연구들은 주로 이미지 프로세싱과 비실시간 처리에 GPU를 적용했을 뿐 실시간 레이다 신호처리에는 적용하지 않고 있다. 따라서 본 논문에서는 우주감시레이다에서 장거리 표적 탐지/추적에 따른 연산시간 증가 문제를 저비용으로 해결하기 위해 HPC 에 GPU의 CUDA (compute unified device architecture) Core를 적용해 실시간 레이다 신호처리를 하는 방법에 대해 제시한다.

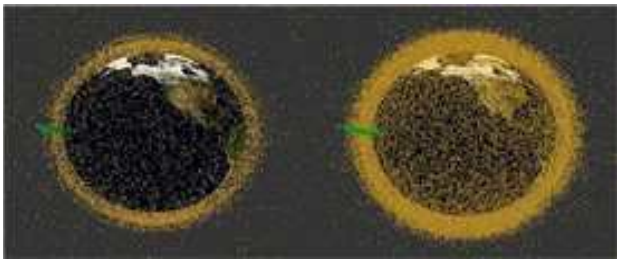

그림 1. 저궤도 우주 물체 카달로그 [9]
Fig. 1. LEO space object catalog [9].

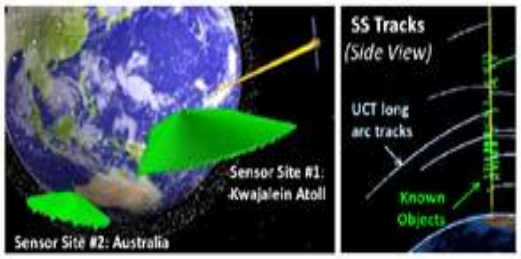

그림 2. 스페이스 펜스 레이다의 탐색 범위와 운용도 [10]
Fig. 2. Space fence radar operation concept [10].

Ⅱ. 우주 물체 감시

2-1 우주 물체 감시 영역과 감시 필요성

우주 물체는 지구로부터 고도상으로 떨어져 있는 위치에 따라 저궤도, 중궤도 등으로 나누어진다. 통신 위성, 감시 위성, 큐브 위성 등 대부분의 위성은 저궤도에 위치하며, 그림 1은 저궤도에서 임무를 수행 중인 위성 및 수명이 다한 위성과 위성으로부터 생긴 잔해물, 우주 쓰레기의 분포를 나타낸다. 우주 물체 감시는 한반도 상공을 지나가는 우주 물체를 탐지/추적하고 이로부터 궤도를 분석하여 DB(database)화하고 카탈로그를 최신화하여 한반도 상공의 우주 물체를 인지하고 우주 쓰레기 및 타 위성으로부터 자국 위성의 충돌 방지 등을 하는데 필요하다. 이러한 우주 물체 감시 수단으로는 광학 카메라와 레이다가 있으며, 광학 카메라는 감시 영역이 협소하고 상시 감시가 불가능하기 때문에 상시 광역 감시를 위해서는 레이다가 효과적이며 대표적인 사례로 미국에서 운용 중인 Space Fence 레이다가 있다. 이러한 Space Fence 레이다는 ±60도의 넓은 방위각을 상시로 탐색 가능하며, 그림 2는 Space Fence 레이다의 탐색 범위 및 운용도를 나타낸 것이다.

2-2 우주 물체 감시를 위해 필요한 레이다 운용

장거리에 존재하는 우주 물체를 감시하기 위해 클러터의 영향을 비교적 덜 받는 LPRF (low pulse repetition frequency) 파형을 사용하며 이는 HPRF(high pulse repetition frequency), MPRF(medium pulse repetition frequency) 파형보다 큰 데이터를 처리해야 한다. 또한 광역 감시로 인해 레이다 탐색 프레임 타임이 증가하는 문제를 해결하기 위해 다중 주파수 빔 운용이 필요하다. 그리고 대기권의 전리층으로 인한 전파 왜곡 위험을 줄이기 위해 이중 편파 수신 기능이 요구된다. 마지막으로 수만 개의 배열 소자로부터 수신받은 신호를 다중 빔으로 형성하는 빔포밍도 수행해야 한다. 앞서 제시한 4가지 기술은 우주 물체 감시를 위해 필요하지만 기존 레이다에 비해 파형의 길이와 데이터 크기가 크기 때문에 이를 효과적으로 처리할 수 있는 방안이 필요하다.

Ⅲ. GPU CUDA

3-1 GPU 연산구조

GPU는 그래픽 처리에 관련된 부동 소수점 연산을 담당한다. 1990년대 중반 게임과 미디어의 인기에 힘입어 엄청난 발전을 해왔다. 실제로 부동 소수점 연산 능력을 비교해 봤을 때 2000년대 중반 이미 그래픽 카드가 인텔 CPU의 연산 성능을 압도하는 것으로 나타났다 [11]. GPU는 그래픽 처리를 하지 않을 때 유휴 상태로 전환된다. 따라서 이 GPU 유휴 자원을 범용 연산에 사용하고자 하는 GPGPU(General Perpose GPU)에 대한 연구가 급속도로 진행되었다 [12]. 그림 3과 같이 다수의 작은 프로세서로 구성된 GPU는 특히 병렬처리에 탁월한 성능을 보인다. 여러 개의 GPU를 탑재한다면 병렬처리 성능은 더욱 향상될 수 있다. 하지만 이런 장점에도 불구하고 GPU 자원을 사용자가 편리하게 사용할 수 있는 범용 API의 부재, 느린 인터페이스 속도, 정수연산 지원 불가 등의 문제로 제한적으로 사용되어 왔다. 그러나 NVIDIA에서 발표한 CUDA 플랫폼과 PCI의 발전으로 위 문제를 대부분 해결하였다 [13]. CUDA는 C/C++, C#, Python 등 일반 프로그래머가 익숙하게 다루는 개발 환경에서 연산을 하듯이 GPU에서 개발을 할 수 있게 지원한다 [14]. 또한, 영상처리나 신호처리에서 주로 사용하는 IPP(integrated performance primitives), MKL(math kernel library), CBLAS(c basic linear algebra subprograms) 라이브러리를 동일한 형태의 내장 함수로 호출할 수 있다. 이에 더해 처리하고자 하는 데이터를 그리드, 블록, 쓰레드로 나누어 사용자가 자유자재로 프로세서를 할당하고 제어할 수 있다 [15]. 그러나 GPU에서 처리해야할 데이터는 CPU로부터 입력으로 받아야하기 때문에 CPU에서 GPU로 데이터를 복사하는 시간, 처리된 데이터를 GPU에서 CPU 로 복사하는 시간은 여전히 연구해야 할 대상이며, 복사 시간을 포함하여 GPU에서 신호처리를 하는 시간이 CPU에서 신호처리 시간보다 빠르다면 적용해 볼 수 있다.

3-2 GPU CUDA 성능

GPU에서 연산 수행 시 그림 4와 같이 데이터 처리 과정이 필요하다. 즉, 주 메모리에서 데이터를 복사해서 GPU 내 메모리로 옮긴 후 GPU를 통해 병렬 연산을 수행하고 다시 주 메모리로 데이터를 복사하는 과정이 필요하기 때문에 이들 소요 시간의 합으로 연산 시간을 정의할 수 있다. 이를 고려하여 CPU와 GPU의 연산 시간 비교를 위해 표 1과 같이 시험 환경을 구성하고, 데이터 량에 따른 CPU와 GPU 간 데이터 복사 시간을 표 2에 정리하였다. 수십 ms의 Dwell Time을 가지는 LPRF 파형에서 10MByte 이하의 데이터가 이동한다면 전체 연산 성능에 영향을 미치는 정도는 적을 것으로 보이지만 샘플링 주기가 커지는 100MByte 크기의 데이터가 이동할 때는 부정적인 영향을 줄 수 있을 것으로 판단된다.

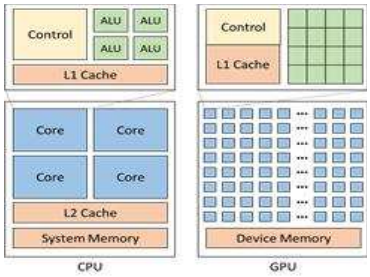

그림 3. CPU와 GPU 구조 비교
Fig. 3. Comparison of structure for CPU and GPU.

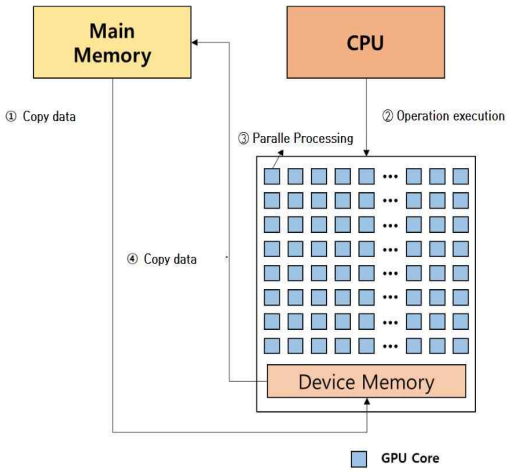

그림 4. GPU 데이터 처리 과정
Fig. 4. GPU Data processing flow.

표 1. GPU와 CPU 시험 환경
Table 1. GPU and CPU test condition.

Item		Detail
CPU	Model	Intel Xeon Gold 6354
	# of Core	18
	Clock	3.0 GHz
	Memory	64 G
GPU	Model	NVIDIA RTX A4000
	# of Core	6144
	Clock	1.5 GHz
	Memory	16 G
OS	RHLE 7 Version	

표 2. GPU ↔ CPU 데이터 복사 시간 비교
Table 2. Comparison of GPU ↔ CPU data copy speed.

Copy data size	CPU → GPU (ms)	GPU → CPU (ms)
100 Byte	0.0506	0.0419
1 KByte	0.0339	0.0336
10 KByte	0.0345	0.0324
100 KByte	0.0329	0.0311
1 MByte	0.1459	0.1488
10 MByte	0.9516	0.9473
100 MByte	9.9037	9.9021

표 3. CPU와 GPU의 FFT 연산 시간 비교
Table 3. Comparison of FFT processing time between CPU and GPU.

FFT Size	CPU(ms)	GPU(ms)
2^{15}	0.0564	0.0561
2^{16}	0.0684	0.0513
2^{17}	0.1548	0.0801
2^{18}	0.2713	0.1310
2^{19}	0.7032	0.2735

표 4. CPU와 GPU의 행렬 연산 시간 비교
Table 4. Comparison of matrix processing time between CPU and GPU.

Item	CPU(ms)	GPU(ms)
+	2.6324	0.1375
-	2.5402	0.1375
*	2.5831	0.1375
/	4.1543	0.2509

또한 레이다 신호처리 수행 시 가장 자주 사용되는 FFT 연산 및 행렬 연산 시간을 CPU 및 GPU에서 각각 산출하여 표 3 및 표 4와 같이 비교하였다. 표 3에서 FFT 연산 시간을 비교하여 살펴보면, FFT 크기가 클수록 CPU와 GPU의 연산 시간 차이가 크게 나타나는 것을 확인할 수 있으며 이는 우주감시레이다와 같이 탐지 거리가 멀어 파형의 길이가 큰 파형일수록 펄스 압축 시 GPU를 적용하면 연산 시간을 크게 단축시킬 수 있음을 의미한다. 표 4는 64 bit형 1024 x 1024 크기의 행렬에 대해 연산자에 따른 연산 시간을 정리한 표로 GPU에서 16~19배 연산이 빠른 것을 확인할 수 있으며, 위 결과로부터 GPU를 적용한 레이다 신호처리 시 데이터 복사 시간을 고려하더라도 연산 시간 단축 효과를 기대할 수 있음을 알 수 있다.

Ⅳ. GPU 적용 신호처리 시험 결과

4-1 GPU를 사용한 신호처리 흐름

LPRF 파형을 사용하는 레이다 신호처리는 일반적으로 신호 수신, 아날로그/디지털 변환, 데이터 정렬, 펄스 압축, 비동기 누적, CFAR, 클러스터링, 모노펄스 순서로 처리된다. 하지만 신호 수신 및 아날로그/디지털 변환 데이터 정렬은 안테나 장치와 GPU 간 직접 연결이 어렵다. 또한 클러스터링은 거리 방향의 모든 데이터 샘플을 차례대로 순차적으로 비교하면서 연산을 진행하기 때문에 병렬처리의 이점을 살리기 어렵다 [16]. 따라서 본 논문에서 GPU를 사용해 적용하고자 하는 알고리즘은 펄스 압축, 비동기 누적, CFAR로 제한하며, 그림 5는 CPU와 GPU 간 데이터 복사를 포함해 GPU를 이용한 레이다 신호처리 흐름도를 나타낸다.

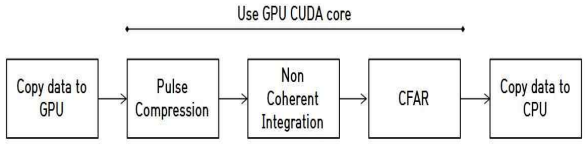

그림 5. GPU를 이용한 레이다 신호처리 흐름도
Fig. 5. Processing flow of radar signal processing with GPU.

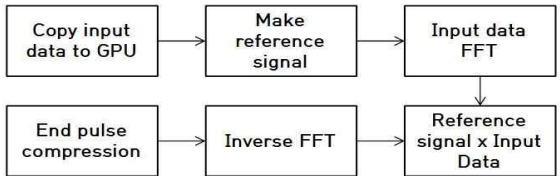

그림 6. 펄스압축 흐름도
Fig. 6. Processing flow of pulse compression.

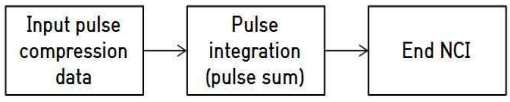

그림 7. 비동기 누적 흐름도
Fig. 7. Processing flow of noncoherent integration.

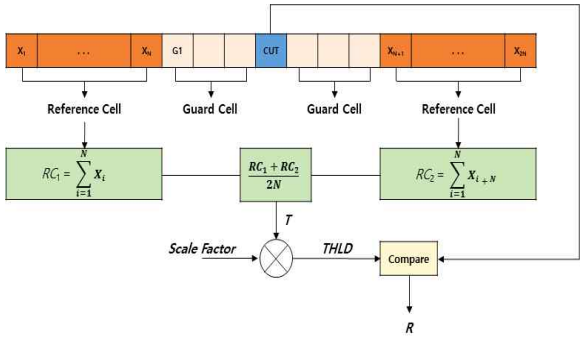

그림 8. CA-CFAR 흐름도
Fig. 8. Processing flow of CA-CFAR.

레이다 신호처리 과정은 펄스 단위 혹은 입력 채널별로 병렬처리를 하였으며 이에 맞게 GPU 코어를 할당했다. 펄스 압축은 입력 신호와 기준 신호의 Convolution으로 상관관계를 계산하는 신호처리로 이는 일반적으로 그림 6과 같이 FFT 및 Inverse FFT로 수행된다. 비동기 누적은 하나의 버스트에서 수신받은 펄스를 누적하여 신호 대 잡음비를 향상시키는 신호처리로 그림 7과 같이 수행되며 연산 상으로는 단순히 행렬 간 덧셈 연산이 수행된다. CFAR(Constant False Alarm Rate)는 잡음 신호를 표적으로 인식해 발생하는 오경보를 일정하게 유지하면서 표적 신호 탐지를 위한 임계값을 설정하는 신호처리로 본 논문에서는 Reference Cell의 평균을 구해 임계값을 만드는 CA(Cell Average) CFAR를 사용하였으며, 이를 위해 각 Reference Cell의 값들에 대한 행렬 간 덧셈 등이 수행되고 이는 그림 8에 나타내었다.

표 5. 레이다 신호처리 입력 데이터
Table 5. Input data for radar signal processing.

Ch	PRI (ms)	Pulse	PW (ms)	BW (Mhz)	Sampling rate (Mhz)	FFT size	Data size (Byte)
O	OO	O	O	O	OO	OOO,OOO	9,216,000

표 6. CPU와 GPU 신호처리 연산 시간 비교
Table 6. Comparison of radar signal processing time between CPU and GPU.

Item	CPU(ms)	GPU(ms)
Copy data CPU to GPU	0	0.4069
Pulse compression	11.1719	2.0922
NCI	2.0912	0.1663
CA-CFAR	7.0296	1.1456
Copy data GPU to CPU	0	0.3934
Total processing time	20.2927	4.2044

4-2 신호처리 시험결과

신호처리 성능을 비교하기 위한 입력 데이터는 안테나 신호 수신, 아날로그/디지털 변환, 데이터 정렬이 끝난 데이터이며, 이 때 파형 정보는 표 5에 나타내었다. 또한 이 때 각 레이다 신호처리 과정에 대한 연산 시간을 CPU 및 GPU 별로 표 6에 정리하였다. GPU를 이용한 신호처리 성능 측정 결과 CPU 신호처리 대비 펄스 압축에서 약 5.33배, 비동기 누적에서 약 16.57배, CA-CFAR에서 약 6.13배 빠른 성능을 보였고 총 4.82배 빠른 성능을 보였다. 특히 비동기 누적에서 16.57배 빠른 연산 시간이 나타났는데, 비동기 누적은 펄스 데이터의 단순 행렬 합으로 표현할 수 있고 행렬의 합에 대한 연산 시간은 표 4의 결과로 나타낸 바 있다. 펄스 압축에서도 FFT와 행렬 연산을 수행함으로 연산 시간이 줄어든 것을 확인할 수 있다.

Ⅴ. 결 론

우주감시레이다의 신호처리에 CPU 및 GPU 적용 시 연산 시간을 비교하였고 그 결과 GPU가 총 4.82배 빠른 연산 속도를 갖는 것을 확인하였다. 특히 펄스 압축과 비동기 누적 연산 시 GPU를 이용한 FFT와 행렬 연산으로 크게 연산 시간을 단축시킬 수 있었다. 기존의 레이다 신호처리장치는 주로 VPX 형태로 구성되어 개발되어 왔기 때문에 GPU 적용이 어려웠으나 최근 진동, 충격 등 환경적인 영향이 적은 응용 분야에서 HPC 형태로 개발되고 있어 GPU 적용이 용이하고 이 때 연산 시간 단축을 기대할 수 있다.

Acknowledgments

본 연구는 대한민국 정부 (산업통상자원부, 과학기술정보통신부 및 방위사업청) 재원으로 민군협력진흥원에서 수행하는 민군기술협력사업의 연구비 지원으로 수행되었습니다. (과제번호 23-CM-RS-12)

References

[1] C. Pardini and L. Anselmo, “Environmental sustainability of large satellite constellations in low earth orbit,” *Acta Astronautica*, Vol. 170, pp. 27-36, May 2020, DOI: https://doi.org/10.1016/j.actaastro.2020.01.016.

[2] T. Schildknecht, “Optical surveys for space debris,” *Astronomy and Astrophysices Review*, Vol. 14, No. 1, pp. 41-111, Jan. 2007, DOI: https://doi.org/10.1007/s00159 − 006-0003-9.

[3] J. Choi, C. Park, Y. Kim, H. Kim, J. Kwon and G. H. Kim, “A design study of signal processor for small tracking radar,” *The Journal of the Institute of Internet, Broadcasting and Communication*, Vol. 20, Issue 5, pp. 71-77, Oct. 2020, DOI: https://doi.org/10.7236/JIIBC.2020.20.5.71.

[4] I. D. Gerg, D. C. Brown, S. G. Wagner, D. Cook, B. N. O'Donnell, T. Benson and T. C. Montgomery, “GPU acceleration for synthetic aperture sonar image reconstruction,” in *Global Oceans 2020: Singapore–US Gulf Coast*, Biloxi: MS, pp. 1-9, Oct. 2020, DOI: https://doi.org/10.1109/IEEECONF38699.2020.9389388.

[5] S. S. Maddikonda and G. A. Shanmugha Sundaram, “SAR image processing using GPU,” in *2014 International Conference on Communication and Signal Processing*, Melmaruvathur: India, pp. 448-452, April 2014, DOI: https://doi.org/10.1109/ICCSP.2014.6949881.

[6] B. Liu, K. Wang, X. Liu and W. Yu, “An efficient SAR processor based on GPU via CUDA,” in *2009 International Congress on Image and Signal Processing*, Tianjin: China, pp. 1-5, Oct. 2009, DOI: https://doi.org/10.1109/CISP.2009.5304418.

[7] D. A. Zherdev, V. A. Procudin, E. Y. Minaev and V. A. Fursov, “HPC implementation of radar images modelling method using CUDA,” *Journal of Physics: Conference Series*, Samara: Russia, p. 012083, April 2018, DOI: https://doi.org/10.1088/1742-6596/1096/1/012083.

[8] R. W. Linderman, J. Corner and S. Tucker, “Swathbuckler: real-time wide swath synthetic aperture radar image formation using embedded HPC,” in *2006 HPCMP Users*

Group Conference, Denver: CO, pp. 244-251, June 2006, DOI: https://doi.org/10.1109/HPCMP-UGC.2006.68.

[9] M. G. Koltiska, B. Pierce, C. Maldonado, T. Quiller, M. McHarg and B. Bishop, "Utilizing cubesatellites for characterization of the AN/FSY-3 space fence system and other sensors," in *Advanced Maui Optical and Space Surveillance (AMOS) Technologies Conference*, Maui: HI, p. 100, Sept. 2017, Retrieved from https://amostech.com/TechnicalPapers/2017/Poster/Koltiska.pdf.

[10] G. Fonder, M. Hughes, M. Dickson, M. Schoenfeld and J. Gardner, "Space fence radar overview," in *2019 International Applied Computational Electromagnetics Society Symposium*, Miami: FL, pp. 1-2, April 2019, Retrieved from https://ieeexplore.ieee.org/abstract/document/8712890.

[11] D. Luebke, M. Harris, N. Govindaraju, A. Lefohn, M. Houston, J. Owens, ..., I. Buck, "GPGPU: general-purpose computation on graphics hardware," in *Proceedings of the 2006 ACM/IEEE conference on Supercomputing*, Tampa: FL, pp. 208-es, Nov. 2006, DOI: https://doi.org/10.1145/1188455.1188672.

[12] J. Lee, "A study of the GPGPU performance," *The Journal of the Institute of Internet, Broadcasting and Communication*, Vol, 18, Issue 6, pp. 201-206, Dec. 2018, DOI: https://doi.org/10.7236/JIIBC.2018.18.6.201.

[13] D. Kirk, "NVIDIA CUDA software and GPU parallel computing architecture," in *Proceedings of the 6th International Symposium on Memory Management*, Montreal Quebec: Canada, pp. 103-104, Oct. 2007, DOI: https://doi.org/10.1145/1296907.1296909.

[14] H. S. Bae, S. H. Yu and H. Y. Kwon, "Data assimilation of real-time air quality forecast using CUDA," *The Journal of the Institute of Internet, Broadcasting and Communication*, Vol, 17, Issue 2, pp. 271-277, April 2017, DOI: https://doi.org/10.7236/JIIBC.2017.17.2.271.

[15] Y. I. Cho, C. L. Ha, J. H. Yang and J. H. Kim, "The design of parallel processing S/W using CUDA for realtime 3D laser ladar imaging system," *Journal of the Korea Society of Computer and Information*, Vol. 18, Issue 1, pp. 1-10, Jan. 2013, DOI: https://doi.org/10.9708/jksci.2013.18.1.001.

[16] A. Bachoo, "Using the CPU and GPU for real-time video enhancement on a mobile computer," in *IEEE 10th International Conference on Signal Processing Proceedings*, Beijing: China, pp. 405-408, Oct. 2010, DOI: https://doi.org/10.1109/ICOSP.2010.5657164.

조 인 철 (In-Cheol Cho)
2015년 2월: 서원대학교 컴퓨터교육과 (공학사)
2017년 2월: 인하대학교 컴퓨터공학과 (공학석사)
2017년 1월 ~ 현재: LIG넥스원 레이다연구소
※관심분야 : 레이다 신호처리, 레이다 통제

조 성 훈 (Sung-Hoon Cho)
2021년 8월: 인하대학교 컴퓨터교육과 (공학사)
2023년 8월: 인하대학교 컴퓨터공학과 (공학석사)
2024년 1월 ~ 현재: LIG넥스원 레이다연구소
※관심분야 : 레이다 신호처리, 레이다 통제

안 지 훈 (Ji-Hoon An)
2014년 2월: 한양대학교 전자시스템공학 (공학사)
2016년 2월: 한양대학교 전자시스템공학 (공학석사)
2016년 1월 ~ 현재: LIG넥스원 레이다연구소
※관심분야 : 레이다 통제, 레이다 추적

문 현 욱 (Hyun-Wook Moon)
약2005년 2월: 연세대학교 전기전자공학과 (공학사)
2007년 2월: 연세대학교 전기전자공학과 (공학석사)
2016년 2월: 연세대학교 전기전자공학과 (공학박사)
2014년 3월 ~ 현재: LIG 넥스원 레이다연구소
※관심분야 : 레이다 성능분석, 전파전파, 무선채널 모델링

손 성 환 (Sung-Hwan Sohn)
2004년 2월: 인하대학교 전자공학과 (공학사)
2006년 2월: 인하대학교 정보통신대학 (공학석사)
2010년 8월: 인하대학교 정보통신대학 (공학박사)
2011년 1월 ~ 현재: LIG 넥스원 레이다연구소
※관심분야 : 레이다 신호처리, 레이다 통제, 레이다 성능분석

정 태 희 (Taehee Jeong)
2009년 2월: 한양대학교 전자통신컴퓨터공학부 (공학사)
2011년 2월: 포항공과대학교 전자전기공학과 (공학석사)
2024년 2월: 한국과학기술원 전기및전자공학부 (공학박사)
2011년 1월 ~ 현재: 국방과학연구소 선임연구원
※관심분야 : 레이다 신호처리, 표적 탐지, 레이다 시스템

임 상 호 (Sangho Lim)
2006년 2월: 중앙대학교 전자전기공학과 (공학사)
2008년 2월: 한국과학기술원 전기및전자공학부 (공학석사)
2011년 8월: 한국과학기술원 전기및전자공학부 (공학박사)
2011년 9월 ~ 2016년 9월: 삼성전자 책임연구원
2016년 10월 ~ 현재: 국방과학연구소 책임연구원
※관심분야 : 영상레이다, 우주감시레이다