https://doi.org/10.7840/kics.2024.49.9.1295

MATLAB의 PCT와 CMEX를 사용한 빠른 획득과 추적을 위한 오픈 소스 GPS L1 C/A SDR 구현

유 승 수◆, 유 재 덕*, 김 선 용°

Open Source GPS L1 C/A SDR Implementation for Fast Processing of Acquisition and Tracking Using MATLAB PCT and CMEX

Seungsoo Yoo◆, Jae Duk Yoo*, Sun Yong Kim°

요 약

본 논문에서는 MATLAB의 PCT (Parallel Computing Toolbox)와 CMEX (C/C++ for Matlab EXcutable) 기능을 이용한 빠른 획득과 추적 처리를 위한 오픈 소스(open source) GPS (Global Positioning System) L1 C/A (Coarse/Acquisition) SDR (Software-Defined-Radio)을 구현하고, 그 성능을 평가한다. 본 논문에서 구현한 SDR은 가독성(readability)과 가변성(flexibility)에 중점을 둔 교육 및 연구용 SDR로서 MATLAB을 주 언어로 구성하였다. [9]의 대표적인 MATLAB 기반 오픈 소스 GPS L1 C/A SDR은 성능 평가를 위해 사용한 컴퓨터에서 약 37초의 IF (Intermidiate Frequency) 표본을 처리할 때, 총 처리 시간은 약 431.301초가 소요되며, 이 가운데 획득 시 약 3.78%인 약 16.296초가 소요되며, 추적 시 91.47%인 약 395.276초가 소요되는 것을 확인하였다. 이를 개선하기 위해 MATLAB에서 제공하는 PCT와 CMEX 기능 등을 활용해 빠른 획득과 추적을 위한 GPS L1 C/A SDR을 구현하였다. 같은 컴퓨터에서 구현한 SDR의 총 처리 시간은 약 73.086초로서 기존 SDR 대비 약 17.19%의 처리 시간만 소요되며, 획득은 기존 SDR의 약 16.296초 대비 약 60.85%인 약 9.916초, 추적은 기존 SDR의 약 395.276초 대비 약 9.05%인 약 35.79초로 빠른 획득과 추적을 할 수 있음을 확인하였다. 구현한 빠른 획득과 추적을 위한 SDR은 관련 교육 및 연구에 쉽게 활용할 수 있도록 [10]과 같이 오픈 소스로 공개한다.

키워드 : 범역측위시스템, 소프트웨어 기반 수신기, 오픈 소스, 매트랩, PCT, CMEX

Key Words : Global Positioning System (GPS), Software Defined Radio (SDR), MATLAB, Open Source, Parallel Computing Toolbox (PCT), and C/C++ for MATLAB EXcutable (CMEX)

ABSTRACT

In this paper, we implement an open source Global Positioning System (GPS) L1 Coarse/Acquisition(C/A) Software-Defined-Radio (SDR) for fast processing of acquisition and tracking using Parallel Computing Toolbox (PCT) and C/C++ for MATLAB EXcutable (CMEX), and evaluate its performance. The SDR implemented in

※ 본 연구는과학기술정보통신부 및 정보통신기획평가원의 대학ICT연구센터사업 (grant no. IITP-2024-RS-2023-00258639)의 연구결과로 수행되었습니다.

◆ First Author : Konkuk University, Department of Electrical and Electronics Engineering, kelvin@konkuk.ac.kr, 정회원

° Corresponding Author : Konkuk University, Department of Electrical and Electronics Engineering, kimsy@konkuk.ac.kr, 종신회원

* Konkuk University, Department of Electronics, Information & Communication Engineering, woejr120@konkuk.ac.kr, 학생회원

논문번호 : 202404-071-B-RU, Received April 19, 2024; Revised May 20, 2024; Accepted May 24, 2024

this paper is an SDR for education and research that focuses on readability and flexibility and is mainly written in MATLAB. In this paper, a representative MATLAB-based open source GPS L1 C/A SDR was implemented in [9], and when the used computer processes IF (Intermediate Frequency) samples of about 37 seconds, the total processing time takes about 431.301 seconds, of which acquisition takes about 16.296 seconds (about 3.78%), and tracking takes about 395.276 seconds (91.47%). To improve this, we implemented an GPS L1 C/A SDR for fast acquisition and tracking using Parallel Computing Toolbox (PCT) and CMEX provided by MATLAB. The total processing time of the proposed SDR on the same computer is about 73.086 seconds, which is only about 17.19% of the processing time of the conventional SDR, and the acquisition is about 9.916 seconds, which is about 60.85% of the 16.296 seconds of the conventional SDR, and the tracking is about 9.916 seconds of 35.79 seconds of the conventional SDR, which is about 9.05% of the 395.276 seconds of the conventional SDR. The proposed SDR for quick acquisition and tracking is released as open source as in [10] so that it can be easily used for related education and research.

Ⅰ. 서 론

KPS (Korea Positioning System)는 한반도 인근 지역에 초정밀 PNT (Position, Velocity, and Timing)를 제공하기 위한 시스템으로서, 2022년부터 2035년까지 KPS 위성 시스템과 지상 시스템, 그리고 사용자 시스템을 개발할 예정이다. KPS 사용자 시스템을 포함한 수신기는 무선 전파를 수신해 디지털 표본을 얻는 아날로그 부분과 디지털 표본 신호를 처리해 원하는 정보를 얻는 디지털 부분으로 구성된다. 아날로그 부분은 수신기 앞단에 있어서 RFFE (Radio Frequency Front-End)라고도 부른다. 일반적인 상용 GNSS (Global Navigation Satellite System) 수신기는 RFFE와 GNSS 신호처리에 적합한 집적회로(integrated circuit)로 구성되며, 가격도 저렴하고, 성능도 우수하다. 그러나 하드웨어 수신기에 신호처리 기능을 변경하거나 추가하려면 새로운 집적회로 설계 및 제작이 필요해서 시간적, 경제적 부담이 상당히 높다. 이에 대한 대안으로 연구 개발 시점에서 디지털 부분을 소프트웨어로 대체해 구성하는 SDR (Software Defined Radio or Receiver)이 많이 활용된다[1].

GNSS SDR은 일반 목적 컴퓨터에서 GNSS 수신 신호를 처리해 PVT를 결정하는 수신기이다. 현재 GNSS SDR은 사용 목적에 따라 1) 실시간 SDR[4,5], 2) 교육 및 연구 개발용 SDR[6-9], 그리고 3) Snapshot SDR, 이상 세 종류로 구분한다. 1) 실시간 SDR은 C/C++과 같은 시스템 프로그래밍 언어로 설계된 SDR로서 실시간 처리가 가능하도록 최적화된 SDR이다. 그러나 실시간 SDR은 그 구조가 복잡하고, 시스템 프로그래밍 언어에 숙련된 개발자만 다룰 수 있어 사용이 제한적이다. 2) 교육 및 연구 개발용 SDR은 프로그래밍에 숙련된 개발자가 아니어도 쉽게 다룰 수 있는 프로그래밍 언어인 Python[6], MATLAB[7-9] 등의 프로그래밍 언어로 설계된 SDR로서 가독성(readability)과 유연성(flexibility)이 높다. 3) Snapshot SDR은 저가 저전력의 사물 인터넷(Internet of Things, IoT) PNT 센서 등에서 사용되는 SDR로서 처리 시간 또는 효율보다는 IoT 내장 처리 장치에서 실행될 수 있도록 구성한 SDR이다[1-3]. 본 논문에서는 연구 개발 중인 KPS 신호의 효율적인 성능 분석 및 수신 처리 기법 개발을 위해 2) 교육 및 연구 개발용 SDR에 초점을 맞춘다.

대표적인 GNSS SDR은 표 1과 같다. 표 1에서 정리한 대표적인 SDR 가운데 성능면에서 가장 우수한 SDR은 GRID이다. GRID는 C++ 언어로 구성되어 있으며, 15년 이상의 지속적인 개발로 가장 성숙한 수준의 실시간 처리가 가능하고, 처리 능력이 다소 낮은 내장형 시스템(embedded system)에서도 SIMD (Single Instruction Multiple Data) 가속화 등을 적용한 최신 신호처리 기법들을 처리할 수 있다. 그러나 오픈 소스(Open Source, OS) 소프트웨어가 아니다. 같은 언어로 구성된 SDR 가운데 OS SDR은 GNSS-SDR[4,5]이다. GNSS-SDR은 GIRD보다 성능은 낮지만, OS SDR 중 가장 처리 속도가 빠르다. 하지만 두 SDR 모두 C++ 언어를 기반으로 구성하여서 숙련된 시스템 프로그래밍 언어 개발자만 사용할 수 있다. SyDR[6]은 OS SDR 중 Python 언어로 구성된 GNSS SDR 가운데 하나로서, Python 언어의 장점인 높은 범용성을 갖지만, 처리 속도가 소개한 SDR 가운데 가장 느리다. AutoNavSDR, SoftGNSS[7], FGI-GSRx[8], OS GNSS SDR Collection[9]은 모두 MATLAB 언어로 구성된 GNSS SDR이다. 이 가운데 AutoNavSDR은 API (Application Programming Interface) 구성 및 GPU (Graphic

표 1. 대표적인 GNSS SDR과 그 특징[1-3]
Table 1. Representative GNSS SDRs and their features[1-3]

name	main language	open source	main focus
GRID	C++	No	Real-time operation of advanced algorithms on embedded devices
GNSS-SDR[4,5]	C++	Yes	Real-time SDR with modular structure and widespread use
SyDR[7]	Python	Yes	A controlled environment for testing new processing algorithms, for bench- marking purposes using Python
AutoNav SDR	C & MATLAB	No	Support for KPS development, API, and GPU
Soft GNSS[7]	MATLAB	Yes	Suite of GNSS SDRs with widespread use and accompanying book
FGI-GSRx[8]	MATLAB	Yes	Multi-GNSS SDR with accompanying book
OS GNSS SDR Collection [9]	MATLAB	Yes	Multi-GNSS SDR Collection with accompanying book

Processor Unit) 가속화 등을 사용할 수 있어서 MATLAB 언어로 구성된 GNSS SDR 가운데 상당히 우수한 처리 성능을 보일 것으로 예상된다. 하지만 세부 소스 코드 등은 아직 공개되어 있지 않다[2,3]. MATLAB 언어로 구성된 GNSS SDR 가운데 OS SDR은 SoftGNSS[7], FGI-GSRx[8], OS GNSS SDR Collection[9]이다. 이 세 SDR은 최초의 OS GNSS SDR로부터 파생된 대표적인 GNSS SDR로서 관련 서적 등 관련 자료가 풍부하며, 공개된 주요 기능 요소의 성능은 유사하다. 본 논문은 이 가운데 OS GNSS SDR Collection[9]을 대표적인 기존 SDR로서 분석하고, 이를 KPS 등 관련 연구 개발에 효과적으로 사용할 수 있도록 MATLAB에서 지원하는 고속 처리 방법을 적용한 GNSS SDR을 구성하고, 이를 [10]과 같이 공개한다.

본 서론에 이어 2장에서는 OS GNSS SDR Collection[9]의 주요 구성과 성능을 분석하고, 이를 최적화한 SDR 구현 결과를 설명한다. 그리고, 3장에서는 같은 GNSS 신호 표본을 사용했을 때 OS GNSS SDR Collection[9]과 이를 바탕으로 주요 구성 요소의 고속 처리를 구현한 SDR의 주요 구성별 처리 속도를 비교 분석하고, 4장에서 주요 특징 등을 정리하고 맺는다.

Ⅱ. 기존 SDR 구성과 이를 최적화한 SDR 구현

OS GNSS SDR Collection[9]을 포함해 일반적인 GNSS SDR의 구성은 그림 1과 같다. 그림 1처럼 일반적인 GNSS SDR은 ① 초기화(initialization), ② 획득(acquisition), ③ 추적(tracking), 그리고 ④ 항법신호처리(navigation processing) 부분으로 구성된다.

그림 1의 ① 초기화 단계에서는 표본화 주파수(sampling frequency, f_S), IF (Intermediate Frequency, f_{IF}), 신호 표본 이진 파일 구성 등 주요 매개변수를 초기화한다. ② 획득 단계에서는 수신받은 신호와 수신기에서 생성한 국소(local) 신호를 교차 상관(cross-correlation)하여 각 GNSS 위성 신호의 유무 및 대략적인 부호 위상(code phase)과 도플러 주파수(Doppler frequency, f_D) 정보를 획득한다. ③ 추적 단계는 되먹임(feedback) 구조로 구성하여, 획득된 신호의 세부 부호 동기와 반송파 주파수 위상을 추적 및 결정한다. 끝으로, ④ 항법신호처리 단계에서는 추적 및 결정된 신호에서 의사거리(pseudorange)와 항법 메시지를 얻고, 이로부터 PNT를 결정한다. 일반적인 PNT를 결정하는 방법은 EKF (Extended Kalman Filter)이다. GNSS SDR의 여러 구성 가운데 ③ 추적 단계의 처리 시간이 전체 처리 시간 가운데 약 90% 내외를 차지할 정도로 가장 오래 걸리며, 구성이 복잡하다. 다음은 ② 획득 단계와 ④ 항법신호처리 단계 순이다[1-3].

② 획득 단계와 ③ 추적 단계에서 가장 많이 사용되는 연산은 수신 신호와 국소 신호의 교차 상관 연산이다. OS GNSS SDR Collection[9]에 적용된 FFT (Fast Fourier Transform) 기반 교차 상관 연산의 세부 구성

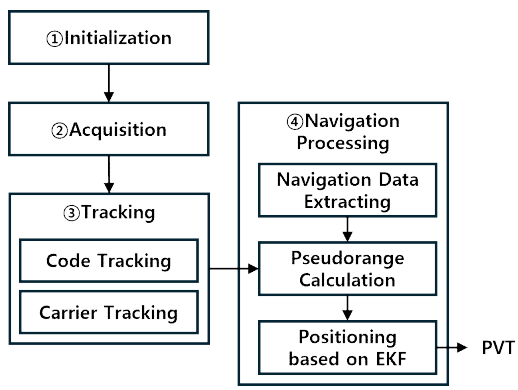

그림 1. 일반적인 GNSS SDR의 구성도
Fig. 1. Block diagram of typical GNSS SDR

도는 그림 2와 같다. 그림 2처럼 교차 상관 연산에서는 먼저 수신된 신호를 수신기에서 설정 또는 예측한 $(f_{IF}+f_D)$의 국소 정현파와 곱하여 IF 대역 신호로 복조하고, 복조된 복소(complex) 신호의 FFT를 수행한다. GNSS 위성 별로 할당된 PRN (Pseudo-Random Noise) 신호는 처리 과정에 따라 미리 또는 필요한 시점에 맞춰 FFT와 결레 복소(complex conjugate) 연산을 취한 후, 복조된 복소 신호의 FFT를 취한 값과 곱하고, 이에 대한 IFFT (Inverse FFT)을 취해 모든 부호 위상에 대한 상관값을 얻는다. 이후 수신 감도(sensitivity)를 높이기 위한 동기(coherent) 또는 비동기(non-coherent) 적산(integration)한 후, 최대 상관 값을 찾고, 이를 미리 설정한 문턱값(threshold)과 비교하여 교차 상관값이 문턱값을 넘는지 판정한다. 그림 2에서 $(\cdot)^*$는 켤레 복소 연산이다.

논문에서 기존 SDR로 활용하는 OS GNSS SDR Collection[9]에 ② 획득 단계에서는 그림 2의 FFT 교차 상관 연산을 사용해 대략적 탐색(coarse search)과 정밀 탐색(fine search) 과정을 수행한다. 본 논문에서는 성능 평가 및 연구 개발에 가장 많이 사용하고 있는 GPS (Global Positioning System) L1 C/A (Coarse/Acquisition) 신호를 적용하였다. OS GNSS SDR Collection[9]의 대략적 탐색에 대한 의사 프로그램 코드 (pseudo-program code)는 그림 3과 같다. 그림 3의 의사 프로그램 코드는 2중 For 루프로 구성되어 있으며, 내부 For 문의 구조는 그림 2의 FFT 기반 교차 상관 연산과 같다. 그림 3처럼 GNSS SDR Collection[9]의 대략적 탐색은 GNSS 위성의 속도와 지상 수신기의 상대적 속도를 고려하여 f_{IF} 기준 ±7kHz 이내의 f_D 범위를 500Hz 간격으로 주파수 탐색 색인 '*freqBinIdx*'에 따라 탐색하며, 탐색을 위한 복소 정현파는 '*sigCarrier*'이다. 본 논문에서는 의사 프로그램 코드 내 변수명 및 함수명은 작은 따옴표 안에 기울임체로 표시한다. 대략적 탐색 과정에는 수신 감도 향상을 위해

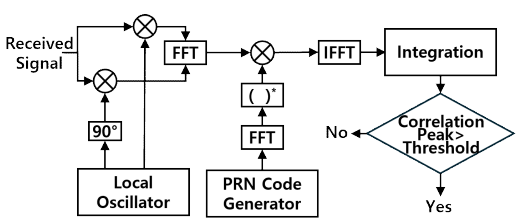

그림 2. OS GNSS SDR Collection[9]에 적용된 FFT 기반 교차 상관 연산의 세부 구성도
Fig. 2. Detailed diagram of FFT-based cross-correlation operation applied to OS GNSS SDR Collection[9]

[Acquisition] Reference [9]. Coarse Search

```
For freqBinIdx = 1 : number of frequency bins (±7kHz, 500Hz step)
    coarseFreqBin(freqBinIdx) = f_IF - 7kHz + 500Hz*(freqBinIdx-1) ;
    sigCarrier = exp(-j*coarseFreqBin*phasePoints) ;

    For ncIdx = 1:acqNcTime (20ms, 2ms step)
        rxSig = longSignal ; (2ms)
        IQfreqDomain = fft(rxSig*sigCarrier) ;
        convCodeIQ = IQfreqDomain .* caCodeFreqDomain ;
        coInt = ifft(convCodeIQ) ;
        ncInt = abs(coInt).^2 ;
        results(freqBinIndex) += ncInt ;
    end
end
```

그림 3. OS GNSS SDR Collection[9]의 대략적 탐색에 대한 의사 프로그램 코드
Fig. 3. Pseudo-program code for coarse search of the OS GNSS SDR Collection[9]

GPS L1 C/A 신호의 20ms 비트 폭을 고려하여 20ms 비동기 적산을 수행한다. 이를 위해 20ms 신호를 2ms 단위로 구분하여 FFT와 IFFT 연산을 반복 수행하여 한 주파수 탐색 색인에 대한 전체 부호 위상을 얻고, 이를 전체 주파수 탐색 범위에 대해 수행하여 한 채널에 대한 대략적 탐색을 위한 상관값을 얻고, 이 가운데 최고값을 미리 설정한 문턱값과 비교해 신호 검출 여부를 판정한다. 대략적 탐색을 통해 신호를 검출한 이후, 그림 3의 대략적 탐색과 유사한 절차로 정밀 탐색을 진행하여 검출된 한 GNSS 위성에 대한 PRN 신호 색인, 부호 위상, f_D 값을 얻으며, 이는 이후 추적 단계에서 해당 PRN 신호의 부호 위상과 반송파 위상 탐색을 위한 초기값으로 사용한다. 대략적 탐색에서는 주파수 탐색 범위를 500Hz로 설정한 것과 비교해 정밀 탐색에서는 대략적으로 탐색을 통해 얻은 f_D 기준 ±250Hz 범위를 25Hz 단위로 탐색한다. 또한, 수신 감도 향상을 위해 GPS L1 C/A 신호의 비트 폭이 20ms인 것을 고려하여 연속된 20ms 길이의 수신신호 2개 블록에 대한 동기 적산을 수행하고, 이 가운데 더 큰 상관값을 갖는 동기 적산 결과로부터 정밀 탐색 결과를 도출한다. 이상과 같이 40ms 길이의 수신 신호에 대한 20ms 단위의 동기 적산을 수행하는 경우, 첫 표본 기준 정확히 10ms에서 비트 천이가 일어나는 경우를 제외한 대부분의 경우에는 최대 약 13dB의 동기 적산 이득을 얻을 수 있다[2,3,9].

본 논문에서 기존 SDR로 활용하는 OS GNSS SDR Collection[9]에 추적 단계에서는 정밀 탐색을 진행하여 검출된 여러 GPS 위성에 대한 PRN 신호 색인, 부호 위상, f_D 값을 바탕으로 GPS C/A PRN 신호 주기와

같은 1ms 길이의 시간 표본에 대한 부호 동기와 f_D의 위상을 추적한다. 추적 단계에서는 현재 추적 중인 부호 위상과 f_D를 사용한 현재(prompt) 시간 영역 교차 상관값을 PLL (Phase Locked Loop)에 입력하여 f_D의 위상을 추적한다. 이때, 사용하는 PLL의 잡음 대역폭은 1.5Hz이며, 감쇠비(damping ratio)는 0.7이다. 부호 동기 위상은 현재 교차 상관하는 부호 위상보다 일정 간격 이른(early) 위상과 늦은(late) 위상을 갖는 상관값을 사용하는 EML-DLL (Delay Locked Loop with Early Minus Late discriminator)을 사용한다. 프로그램 소스 코드로 구현한 EML-DLL의 잡음 대역폭은 1.5Hz이며, 감쇠비는 0.7이다. OS GNSS SDR Collection[9]은 사후 처리 방식의 SDR로서, PNT를 얻기 위해 37초 이상의 신호에 대한 추적이 필요하며, 이전 1ms 추적 결과를 바탕으로 다음 PLL과 EML-DLL에서 새로운 1ms 길이의 신호에 해당하는 표본 수 등을 결정해야 한다. 즉, 이전 1ms 길이 신호의 추적 결과를 다음 1ms 길이 신호의 추적을 위해 되먹임하는 구조로 구성되어 있다. 그러나 서로 다른 PRN을 갖는 신호의 경우, 서로 다른 채널로 독립되어 있으므로, 이를 병렬 처리할 수 있다.

비공개 GNSS SDR을 포함한 주요 SDR의 획득 및 추적 최적화 방법은 표 2와 같다. 표 2처럼 실시간 처리 등 고속 처리를 하려면 C/C++ 언어와 같은 시스템 프로그래밍 언어로 주 구성 요소 또는 전체를 구성해야 한다. 그리고 고속 획득 처리를 위해 최적화된 FFT 기법을 적용하였으며, 고속 추적 처리를 위해 SIMD, 다중 상관기, 그리고 MEX (Matlab EXcutable) 기반 상관기 등을 적용하였다. 그러나 앞서 언급한 것처럼 C/C++ 언어와 같은 시스템 프로그래밍 언어를 주 프로그래밍 언어로 사용하면 연구 개발 난이도가 높아진다. 이를 고려하여 본 논문에서는 MATLAB 기반의 대표적인 OS GNSS SDR 가운데 하나인 OS GNSS SDR Collection[9]으로 주요 요소를 구성하고, MATLAB에서 제공하는 PCT (Parallel Computing Toolbox)와 CMEX (C/C++ for Matlab EXcutable)을 사용해 획득부와 추적부를 고속 처리할 수 있도록 SDR을 구성한다. PCT는 MATLAB에서 병렬 처리를 수행하기 위해 사용하는 함수 모음으로, 여러 코어(core)에서 코드를 실행해 계산 속도를 높일 수 있다. 주요 기능은 병렬 For 루프, 분산 작업자, GPU 컴퓨팅 등이다. PCT를 사용하면 MATLAB 코드의 계산 속도를 빠르게 할 수는 있지만 PCT 처리 블록과 다른 처리 블록의 인터페이스 구성이 다소 복잡하다. CMEX는 MATLAB에서 다양한 C/C++ 코드를 사용할 수 있는 기능으로서, 일정 수준 이상의 적절한 코드가 있다면 수월하게 처리 성능을 높일 수 있다. 그러나 PCT에 비해 구현 난이도가 다소 높고, 적절한 쓰레드(thread) 처리가 필요하다[1-3].

표 2. 주요 SDR의 획득 및 추적 최적화 사례[1-3]
Table 2. Acquisition and tracking optimization cases of representative SDRs[1-3]

	MusNAT	GNSS-SDR	AutoNavSDR
type	real-time	real-time	research/study
language	C/C++, CUDA	C/C++	MATLAB, C
acquisition optimization	optimized FFT	FFT based acquisition	GPU based acquisition
tracking optimization	SIMD	Multicorrelator tracking	MEX correlator7

본 논문에서 최적화한 획득 단계의 세부 구성과 구성도는 각각 표 3과 그림 4와 같다. 표 3과 그림 4처럼 획득 단계는 OS GNSS SDR Collection[9]의 획득부를 'reference'로 두고, 그림 4에서 굵은 붉은색 글씨 등으로 강조한 요소들을 추가하였다. 세부적으로 'reference'와 'configure 1'은 동일한 기법으로 구성하였으며, 기반 언어를 MATLAB에서 C 언어 기반의 CMEX 형식으로만 변경하였다. 획득 단계의 기반 언어를 CMEX로 변경하는 과정에서 MATLAB 데이터 형태로 전달받은 매개변수를 알맞은 형태로 변환하는 작업이 추가로 필요하며, FFT 연산은 외부 라이브러리 FFTW를 통해 수행했다. [10]을 통해 공개한 'configure 1' 프로그램 코드에서 외부 라이브러리 FFTW 함수는 '*fftw_excute( )*' 함수이다. 그리고

표 3. 본 논문에서 최적화한 획득 단계의 세부 구성
Table 3. Detailed configuration of optimized acquisition part

class	language	optimization	
reference	MATLAB	-	
configure 1	MATLAB + CMEX	FFT	Yes
		BLAS	-
		parallel processing	-
configure 2	MATLAB + CMEX	FFT	Yes
		BLAS	Yes
		parallel processing	-
configure 3	MATLAB + CMEX	FFT	Yes
		BLAS	-
		parallel processing	Yes

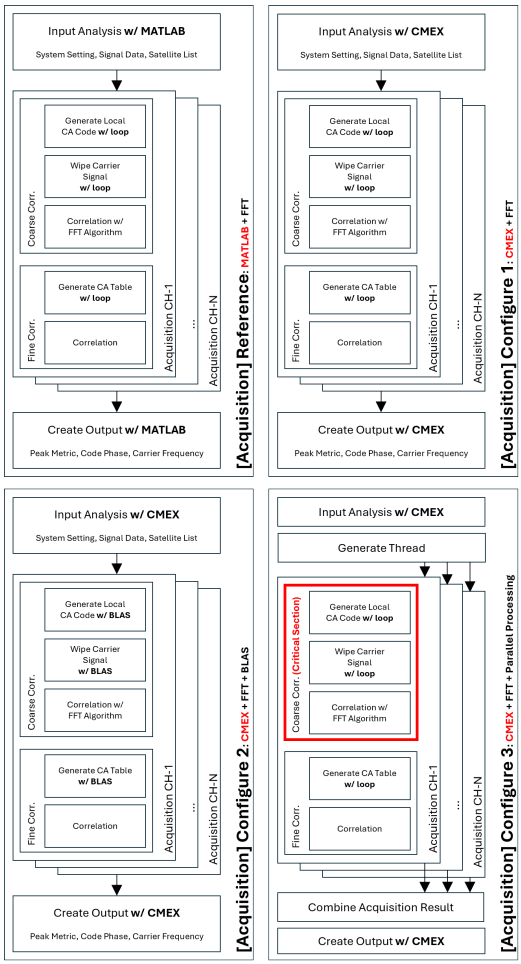

그림 4. 기존 및 구성한 획득 단계의 세부 구성도
Fig. 4. Detailed block diagram of referenced and configured acquisition part

MATLAB 코드를 CMEX 코드로 변경하는 과정에서 그림 3에서 제시한 '*sigCarrier*' 등 행렬 형태의 변수를 사용해 수행되는 요소별 연산을 배열과 루프 형태로 재구성하였다. [10]을 통해 공개한 코드의 대략적 탐색에서는 '*sigCarrReal*'과 '*sigCarrImag*'로 '*sigCarrier*'를 구분하여 해당 색인의 정현파를 생성하였으며, 복조된 신호를 '*buffer*' 변수에 임시 저장한 후, FFT를 수행하고, 이를 GPS L1 C/A 신호 PRN의 FFT 이후 결레 복소 처리한 신호와 곱한 후, IFFT를 수행해 '*corr*' 변수에 저장한 후, 앞서 임시 저장한 비동기 적산을 위해 '*buffer*' 변수에 적산 처리하였다. 정밀 탐색에서 사용한 변수명과 기능도 이와 유사하게 설정하고, 사용하였다. 이 버전에서는 MATLAB에서 사용하는 변수와 CMEX에서 사용하는 변수의 원활한 처리를 위한 전처리 및 후처리 절차가 필요하며, 그림 4 우측 상단의 각 채널별 총 32회 반복 수행하여 획득 단계의 결과를 도출하였다.

'configure 2'는 BLAS (Basic Linear Algebra Subprograms)를 통해 획득 과정에서 선형대수 연산을 최적화할 수 있도록 구성했다. BLAS 라이브러리는 CMEX에서 제공하는 mxBLAS 라이브러리를 사용했다. 이렇게 구성한 이유는 'configure 1'을 구성한 후 세부 시행 결과를 분석하여 MATLAB의 행렬 연산을 CMEX 구성에서 배열과 루프 형태에 의한 추가 실행 시간이 발생함을 확인하였다. 추가 실행 시간의 세부 분석을 위해 'configure 1'의 곱셈, 덧셈 등 일부 루프를 행렬 연산에 최적화된 CMEX에서 제공하는 BLAS 함수를 통해 복조 부분과 정밀 탐색을 위한 C/A 부호 생성 부분을 재구성하였으며, BLAS 함수를 통해 재구성한 복조 부분의 의사 프로그램 코드는 그림 5와 같다. 그림 5 상단의 의사 프로그램 코드는 'configure 1'에서 복조를 위한 정현파를 생성하는 부분이며, 이는 그림 5 하단의 'configure 2'의 '*zaxpy( )*' 함수와 같이 BLAS 라이브러리 함수를 사용해 처리하였다. 이 함수는 입력 x에 가중치 a를 곱한 후, 편이 y를 더해 'ax+y'를 출력하는 함수이다. 그림 5 하단에서 'x'는 신호 표본인 '*sigSample*', 'a'는 '*coarseFreqBin + freqIdx*'을 사용해 정현파를 생성한다. '*one*'과 '*zero*'는 모든 성분이 1 또는 0인 벡터이다. 이후 이를 신호와 곱해 복조하기 위해 그림 5 하단의 '*dsbmv( )*' 함수를 사용하였다. 이 함수는 SMVP (Scalar-Matrix Vector Product)를 수행하는 함수이다.

끝으로, 'configure 3'는 획득 단계에서 사용되는 대

[Acquisition] Configure 1. Remove Carrier in CMEX

```
01: For
02:     ...
03:     double sigCarrReal = cos(coarseFreqBin * sigSample) ;
04:     double sigCarrImag = sin(coaseFreqBin * sigSample) ;
05:     ...
06: end
```

[Acquisition] Configure 2. Remove Carrier w/ BLAS

```
01: zaxpy(&sampleCount, coarseFreqBin+FreqIdx, sigSample, &one, ...
02:     , sigCarr, &one) ;
03: dsbmv("L", &sampleCount, &zero, &one_d, sigCarr, &one, sigIdx, ...
04:     , &one, &zero_d, tes, &one) ;
```

그림 5. 구현한 'configure 1'과 'configure 2'의 복조에 대한 의사 프로그램 코드
Fig. 5. Pseudo-program code for demodulation of implemented 'configure 1' and 'configure 2'

부분의 연산이 루프 사이의 종속성이 없다는 것을 고려하여, 각 PRN 채널별 획득 단계를 병렬로 진행할 수 있도록 구성하였다. 예를 들어, f_S는 18MHz, f_{IF}는 20kHz인 GPS L1 C/A 신호 표본 중 길이 20ms에 해당하는 360,000개의 행렬 연산을 위해 1개의 쓰레드로 360,000개의 표본을 모두 연산하는 속도보다 이를 20개의 쓰레드 나누고, 각 쓰레드에서 18,000개의 표본을 병렬로 연산하는 속도가 더 빠르다. 다만 FFT를 위해 활용한 FFTW의 쓰레드 안정성(thread safety)이 보장되지 않으므로 반드시 직렬로 수행해야 하는 일부 영역을 그림 4 우측 하단에 적색 사각형으로 표시한 부분처럼 주요 직렬 작업 영역(critical serial processing session)으로 구분 및 지정하여 구현하였다. 병렬화 과정에서 주요 직렬 작업 영역에 의한 처리 순차 관리의 복잡성을 고려하여 '*#pragma*'를 통해 영역을 지정해 컴파일러가 자동으로 지정된 영역에 대한 처리 순치 관리가 가능한 컴파일러(compiler)인 OpenMP을 사용하였다. 이를 적용한 의사 프로그램 코드는 그림 6과 같다. 그림 6처럼 01째 줄의 '*#progma omp parallel for*'를 사용하여 다음 루프 영역을 OpenMP를 활용해 병렬로 처리할 것을 지정하였으며, 11째 줄의 '*#progma omp critical*'을 통해 다음 영역을 주요 직렬 작업 영역으로 지정하여 순서대로 처리할 것을 지정하였다. 해당 주요 직렬 작업 영역 내에는 복조 후 13째 줄의 교차 상관을 위한 '*fftw_execute( )*' 함수가 포함되어 있다. 이를 통해 각 PRN 채널에 대한 획득 연산이 병렬로 수행된다. 단, 대략적 탐색 과정에서의 주요 직렬 작업 영역은 병렬 처리 순서대로 순차적으로 처리한다. 이후 설명한 구현 및 성능 분석을 위해 사용한 컴퓨터의 경우, 32개의 PRN 채널에 대해 최대 20개의 쓰레드에 연산을 수행하였다. 이를 통해 더 좋은 성능의 컴퓨터를 사용하여 최대 32개 이상의 쓰레드 또는 채널 수와 같거나 그 이상의 쓰레드를 할당할 수 있는 경우, 수행 속도를 더 높일 수 있음을 예상할 수 있다.

[Acquisition] Configure 3. CMEX+FFT+OpenMP

```
01:  #pragma omp parallel for
02:      for(PRNIdx=0; PRNIdx<MaxIdx; PRNIdx++){
03:          for(freqIdx=0; freqIdx<noOfFreqBins; freqidx++){
04:              double results[36000] = {0, } ;
05:              coarseFreqBin[freqIdx]=f_IF + acqSearchBand ...
06:                      - acqStep*freqIdx ;
07:              for(int ncIdx=0; ncIdx<acqNcTime; ncIdx++){
08:                  const signed char *sigIdx = ...
09:                      dataIdx + ncIdx *2 * samplePerCode;
10:                  ...
11:                  #pragma omp critical // carrier wipe-off
12:                  {
13:                      ...
14:                      fftw_execute(p2);
15:                      ....
16:                  }
17:              }
18:          }
19:      }
20:  }
```

그림 6. 구현한 'configure 3'의 병렬 처리를 위한 의사 프로그램 코드
Fig. 6. Pseudo-program code for parallel processing of implemented 'configure 3'

본 논문에서 최적화한 추적 단계 세부 구성과 구성도는 각각 표 4와 그림 7과 같다. 표 4와 그림 7처럼 추적부는 OS GNSS SDR Collection[9]의 추적부를 'reference'로 두고, 그림 7에서 강조한 요소들을 추가하였다.

'configure 1'은 MATLAB 내에서 각 위성에 대한 추적 연산 병렬 처리를 위해 그림 8처럼 MATLAB에서 제공하는 병렬 처리 요소인 PCT의 '*parfor*'를 01째 및 03째 줄과 같이 적용하였다. 이를 통해 각 PRN 채널의 추적 루프를 MATLAB에서 병렬로 처리할 수 있다. '*parfor*' 문은 병렬 For 루프를 구성하는 문장으로서 각 병렬 루프에 컴퓨터의 코어 또는 워커(worker)를 할당하여 기존 반복문을 처리한다. 단, 기존 for 루프와 같이 순서대로 처리하지 않고, 계산 자원 할당 여부에 따라 비순차적으로 처리한다.

'configure 2'는 'reference'와 같은 처리 알고리즘으로 구성하였으며, 관련 코드를 MATLAB 코드에서 CMEX 코드로 변경하였다. 'configure 2'의 핵심 코드 구성은 그림 9와 같다. 그림 9에서 01째 줄의 '*CHNNEL_MAX*'는 최대 추적 채널 수이며, 병렬 처리를 위한 쓰레드 매개변수의 전달을 위한 '*ThreadParameter*' 구조체를 사전에 정의하고, 07째 및 08째 줄의 '*param*'처럼 사용하였다. '*param*'을 통해 전달 되는 주요 값 가운데 '*chnl*'은 PRN 번호, 부호

표 4. 본 논문에서 최적화한 추적 단계의 세부 구성
Table 4. Detailed configuration of optimized tracking part

class	language	optimization	
reference	MATLAB	-	
configure 1	MATLAB	parallel processing	Yes
configure 2	MATLAB + CMEX	parallel processing	-
configure 3	MATLAB + CMEX	parallel processing	Yes

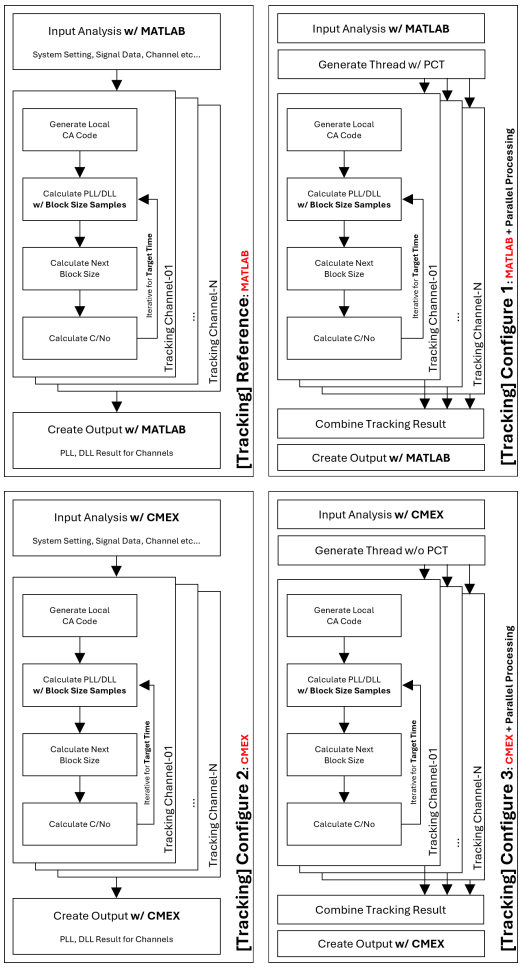

그림 7. 기존 및 구성한 추적 단계의 세부 구성도
Fig. 7. Detailed block diagram of referenced and configured tracking part

[Tracking] Configure 1: MATLAB+PCT

```
01: Parfor chNum = 1 : numberOfChannles
02:     trackResults(chNum) = chNumCalc(...);
03: end
```

그림 8. 구현한 'configure 1'의 PCT 기반 추적 루프에 대한 의사 프로그램 코드
Fig. 8. Pseudo-program code for the PCT-based tracking loop implemented in 'configure 1'

위상 등 해당 채널 신호를 추적하기 위한 주요 정보이며, '*dataInfo*'는 추적하려는 신호 표본의 시작 포인터이다. 이후 '*param*'을 통해 전달받은 주요 매개 변수를 활용하여 '*calcChannelData( )*' 함수 포인터를 사용해 추적 루프를 CMEX 코드로 처리한다. CMEX 코드는 사전 컴파일을 통해 실행 파일로 처리된 후 일괄 실행되므로 PCT에 비해 실행 속도가 빠르다. 다만, 획득 단계

[Tracking] Configure 2: MATLAB+CMEX

```
01: For(int i=0; i<CHANNEL_MAX; i++){
02:     int idx = i ;
03:     result[idx] = initResult(setting);
04:     ThreadParameter param = {&result, idx, *(chnl+idx), ...
05:         setting, dataInfo};
06:     calcChannelData(&param);
07: }
```

그림 9. 구현한 'configure 2'의 CMEX 기반 추적 루프에 대한 의사 프로그램 코드
Fig. 9. Pseudo-program code for the CMEX-based tracking loop implemented in 'configure 2'

의 코드와 유사하게, 전달받은 MATLAB 데이터를 알맞은 형태로 변환하는 과정이 필요하다.

'configure 3'는 'configure 1'의 PCT 대신 병렬 처리 CMEX 기반 추적 루프를 그림 10처럼 구현하였다. 이 구성의 가장 큰 특징은 그림 10의 01째 줄처럼 추적하고자 하는 모든 PRN 채널에 대한 쓰레드 핸들(handle)을 생성하고, 이를 그림 10의 09-10째 줄처럼 각 핸들에 대한 쓰레드를 생성하고, 각 쓰레드, 즉 PRN 채널별로 병렬 추적을 수행한 후, 할당된 모든 PRN 채널에 대한 추적이 완료될 때까지 '*WaitForMultipleObject( )*' 함수를 사용해 처리 및 대기한 후, 추적 결과를 반환한다는 점이다.

[Tracking] Configure 3. M.+CMEX+Parallel Processing

```
01: HANDLE thread[CHANNEL_MAX];
02: Unsigned int threadID[CHANNEL_MAX];
03:
04: For(int i=0; i<CHANNEL_MAX; i++){
05:     int idx = i ;
06:     result[idx] = initResult(setting);
07:     ThreadParameter param = {&result, idx, *(chnl+idx), ...
08:         setting, dataInfo};
09:     thread[idx] = (HANDLE)_beginthreadex(NULL, 0, ...
10:         calcChannelData, (void*)&param, 0, threadID+idx);
11:     sleep(10);
12: }
13: WaitForMultipleObjects(CHANNEL_MAX, thread, TRUE, INFINITE);
```

그림 10. 구현한 'configure 3'의 병렬 처리 CMEX 기반 추적 루프에 대한 의사 프로그램 코드
Fig. 10. Pseudo-program code for the parallel processed CMEX-based tracking loop implemented in 'configure 3'

Ⅲ. 실험 및 성능 분석

논문에서는 기존 및 구성한 빠른 획득 및 추적 처리 성능을 확인하기 위해 OS GNSS SDR Collection[9]에서 얻을 수 있는 37초 이상의 실제 GPS L1 C/A IF 표본을 사용해 각 구성에 따른 성능을 비교하였다. 사용

한 IF 표본은 8비트 동위상(in-phase) 및 직교위상(quadrature) 표본이며, IF는 20kHz, 표본화 주파수는 18MHz이다. 성능 비교를 위해 사용한 시스템의 상대 성능은 그림 11과 그림 12와 같다. 그림 11과 같이 성능 비교를 위해 사용한 컴퓨터는 상대 속도가 가장 빠른 Windows 11 운영체제 기반에 2.4GHz로 동작하는 Intel Core i9-12900와 비교해 약 55%로 약 45%p 처리 능력이 낮다. 그림 12와 같이 그림 11에서의 상대 속도가 가장 빠른 컴퓨터의 비희소 행렬에 대한 상삼각행렬 U와 하삼각행렬 L 분해 연산에 대해서는 약 156%의 시간이 소요되며, FFT에 대해서는 약 174%의 시간이 소요된다.

사용한 시스템에서 실행한 OS GNSS SDR Collection의 주요 블록별 처리 시간은 표 5와 같다. 표 5처럼 총 처리 시간은 약 431.301초이며, 이 가운데 ① 초기화 단계를 처리하기 위해 총 처리 시간 중 약 1.91%에 해당하는 약 8.225초가 소요되며, ② 획득 단계는은 약 3.78%인 약 16.296초, ③ 추적 단계는 약 91.46%인 약 395.276초, 그리고 ④ 항법신호처리 단계에는 약 2.85%인 약 11.504초가 각각 소요되었다. 본 논문에서는 이를 확인하고, 우선 추적부 최적화에 초점을 맞춰 연구를 진행하였으며, 일정 수준 이상의 최적화 이후,

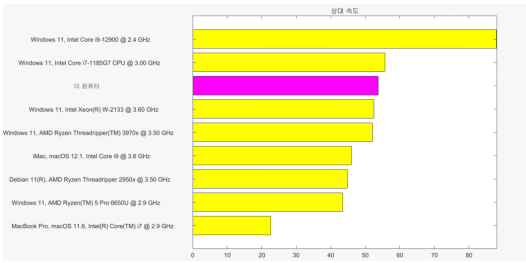

그림 11. 성능 비교를 위해 사용한 컴퓨터의 상대 속도 비교
Fig. 11. Relative speed benchmark of computer used for performance comparison

컴퓨터 유형	LU	FFT
Windows 11, Intel Core i9-12900 @ 2.4 GHz	0.2412	0.1537
Windows 11, Intel Core i7-1185G7 CPU @ 3.00 GHz	0.5330	0.3032
이 컴퓨터	0.3762	0.2681
Windows 11, Intel Xeon(R) W-2133 @ 3.60 GHz	0.3983	0.2755
Windows 11, AMD Ryzen Threadripper(TM) 3970x @ 3.50 GHz	0.2049	0.1730
iMac, macOS 12.1, Intel Core i9 @ 3.6 GHz	0.3410	0.2640
Debian 11(R), AMD Ryzen Threadripper 2950x @ 3.50 GHz	0.3386	0.2451
Windows 11, AMD Ryzen(TM) 5 Pro 6650U @ 2.9 GHz	0.6737	0.3595
MacBook Pro, macOS 11.6, Intel(R) Core(TM) i7 @ 2.9 GHz	0.7366	0.5500

그림 12. 사용한 컴퓨터의 주요 기법별 처리 속도 비교
Fig. 12. Benchmarks of processing speeds by representative processing algorithms

획득부를 최적화하였다. 이상의 설명과 표 5의 ①-④는 그림 1의 일반적인 SDR의 주요 구성요소와 같다.

표 3과 그림 4-6에서 정리한 신호 획득을 위한 네 구성의 성능 비교를 위해 37초 이상의 IF 표본 가운데 20ms 길이의 360,000 표본을 비동기 적산에 활용해 부호 동기 획득 후, 40ms 길이의 720,000 표본을 동기 적산하였다. 그 결과는 표 6과 같다. 표 6처럼 'reference'의 수행 시간은 약 16.296초이며, 'configure 1'과 'configure 2'의 수행 시간은 각각 약 31.531초와 30.566초로, 오히려 'reference'의 수행 시간보다 처리 속도가 약 2배 내외로 느려졌다. 이를 통해 고속 획득 처리를 위해서는 빠른 FFT와 BLAS 기법을 사용하는 것보다는 병렬 처리가 효과적임을 예상할 수 있었으며, 'configure 3'는 약 9.916초로, 'reference'에 비해 획득 시간이 약 60.849%로 약 39.151%p 감소하였다.

표 4와 그림 7-10에서 정리한 추적을 위한 네 구성의 성능 비교를 위해 37초 이상의 IF 표본을 사용해 추적한 결과는 표 7과 같다. 'reference'의 추적 처리 시간은 약 395.276초이며, PCT를 적용한 'configure 1'의 추적 처리 시간은 약 227.132초로 'reference'와 비교해 약 42.539%p 처리 속도가 개선되었다. CMEX만 적용한 'configure 2'의 추적 처리 시간은 약 285.203초로 'reference'와 비교해 약 27.842%p 처리 속도가 개선되었다. 하지만 'configure 1'의 PCT 기반 병렬 처리와

표 5. 사용한 시스템에서 실행한 OS GNSS SDR Collection[9]의 주요 블록별 처리 시간
Table 5. Processing time for each major block of the OS GNSS SDR Collection[9] run on the system used

block	processing time (sec.)	proportion (%)
① initialization	8.225	1.91
② acquisition	16.296	3.78
③ tracking	**395.276**	**91.46**
④ navigation processing	11.504	2.85
total	431.301	100.00

표 6. 본 논문에서 최적화한 SDR의 획득 처리 시간
Table 6. Acquisition processing time of SDR optimized in this paper

class	processing time (sec.)	proportion (%)
reference	16.296	100.000
configure 1	31.531	193.489
configure 2	30.566	187.567
configure 3	**9.916**	**60.849**

표 7. 본 논문에서 최적화한 SDR의 추적 처리 시간
Table 7. Tracking processing time of SDR optimized in this paper

class	processing time (sec.)	proportion (%)
reference	395.276	100.000
configure 1	227.132	57.461
configure 2	285.203	72.153
configure 3	**35.788**	**9.042**

비교해 'configure 2'는 약 125.57%로 더 긴 처리 시간이 소요됨을 확인할 수 있다. 이를 통해 CMEX와 병렬 처리의 효과를 함께 적용하는 것이 가장 빠른 처리 속도를 구현할 수 있음을 확인하고, 'configure 3'처럼 CMEX 구성의 추적 기법을 병렬 처리할 수 있도록 구성하였다. 이처럼 구성한 'configure 3'의 추적 처리 시간은 약 35.788초로, 'reference'에 비해 처리 시간이 약 90.958%p 개선된 것을 확인하였다. 즉, 'configure 3'와 같이 구성하면 37초의 IF 표본을 추적하는데 약 35.788가 소요되어 MATLAB+CMEX 병렬 처리 구조를 사용해 C/C++ 기반 GNSS SDR처럼 실시간에 가까운 처리가 가능함을 확인하였다.

Ⅳ. 결 론

본 논문에서는 빠른 획득과 추적 처리를 위한 OS GPS L1 C/A SDR을 구현하고, 같은 컴퓨터에서 처리 속도를 비교 분석하였다. 본 논문에서 구현한 GNSS SDR은 가독성과 가변성에 중점을 둔 교육 및 연구용 SDR로서 MATLAB을 기반으로 MATLAB에서 지원하는 CMEX 병렬 처리 요소를 추가하였다. 본 논문은 이 가운데 대표적인 MATLAB 기반 오픈 소스 GPS L1 C/A SDR을 구성하고, 사용한 컴퓨터에서 37초의 IF 표본을 처리할 때, 총 처리 시간은 약 431.301초가 소요되며, 이 가운데 획득 시 약 3.78%인 약 16.296초가 소요되며, 추적 시 91.47%인 약 395.276초가 소요되는 것을 확인하였다. 이를 개선하기 위해 MATLAB에서 제공하는 PCT와 CMEX 기능을 활용해 빠른 획득과 추적을 위한 OS GPS L1 C/A SDR을 구현하였다. 같은 컴퓨터에서 구현한 SDR의 총 처리 시간은 약 73.086초로서 기존 SDR 대비 약 17.19%의 처리 시간만 소요되며, 획득은 기존 SDR의 약 16.296초 대비 약 60.85%인 약 9.916초, 추적은 기존 SDR의 약 395.276초 대비 약 9.05%인 약 35.79초로 빠른 획득과 함께 실시간 추적이 가능함을 확인하였다. 구현한 빠른 획득과 추적을 위한 SDR은 논문을 작성하는 2024년 상반기 현재, 공개된 바 없다. 본 논문의 저자들은 독자들이 관련 교육 및 연구에 쉽게 활용할 수 있도록 [10]과 같이 공개한다. MATLAB에서 CMEX의 FFT를 위해 활용하는 FFTW 라이브러리는 병렬 처리를 지원하지 않는다. 따라서 각 프로세스 간 경쟁 상태를 방지하기 위해 쓰레드의 실행 시간이 서로 겹치지 않게 실행되도록 구성하여 각 세부 처리의 동기를 유지했다. 이를 개선하기 위해 FFTW 라이브러리 대신 Intel MKL (Mathematical Kernal Library) 등을 사용해 획득부의 처리 속도를 개선할 수 있을 것으로 예상된다. 추후 이와 함께, [10]을 통해 공개한 프로그래미 코드에 대한 지속적인 의견을 수렴하고, 반영하여 개선된 빠른 SDR 프로그램 코드를 계속 제공할 예정이다.

References

[1] Y.-J. Song, H. Lee, and J.-H. Won, "Design of multi-constellation and multi-frequency GNSS SDR with fully reconfigurable functionality," *JPNT*, vol. 10, no. 2, pp. 91-102, Jun. 2021. (https://doi.org/10.11003/JPNT.2021.10.2.91)

[2] T. Pany, D. Akos, J. Arribas, M. Z. H. Bhuiyan, P. Closas, F. Dovis, I. Fernandez-Hernandes, C. Fernández-Pradez, S. Gunawardena, T. Humphreys, Z. M. Kassas, J. A. L. Scalcedo, M. Nicola, M. L. Psiaki, A. Rügamer, Y.-J. Song, and J.-H. Won, "GNSS software-defined radio: History, current developments, and standardization efforts," *J. The Inst. Navigation: Navigation*, vol. 71, no. 1, pp. 1-45, Spring 2024. (https://doi.org/10.33012/navi.628)

[3] D. Akos, J. Arribas, M. Z. H. Bhuiyan, P. Closas, F. Dovis, I. Fernandez-Hernandez, C. Fernández-Pradez, S. Gunawardena, T. Humphreys, Z. M. Kassas, J. A. L. Scalcedo, M. Niclola, T. Pany, M. L. Psiaki, A. Rügamer, Y.-J. Song, and J.-H. Won, "GNSS software-defined radio: History, current developments, and standardization efforts," in *Proc. 35th Int. Tech. Meeting of the Satellite Division of The Inst. Navigation (ION GNSS+ 2022)*, pp. 3180-3209, Denver. CO, Sep. 2022.

(http://dx.doi.org/10.33012/2022.18434)

[4] C. Fernández-Pradez, J. Arribas, M. Majoral, L. Esteve, P. Closas, J. Vilà-Valls, and D. Miralles, “Centre tecnològic de telecommunications de catalunya,” *An Open-Source Global Navigation Satellite Systems software-defined receiver (GNSS-SDR)(2016)*, Retrieved Apr. 11, 2014, from https://gnss-sdr.org/

[5] C. Fernández-Pradez, *GNSS-SDR, an Open-Source Software-Defined GNSS Receiver*, Retrieved Apr. 11, 2014, from https://github.com/gnss-sdr/gnss-sdr

[6] A. Grenier, *SyDR: An Open-Source Software Defined Radio (SDR) for GNSS Processing in developed in Python*, Retrieved Apr. 11, 2014, from https://github.com/aproposorg/sydr

[7] Z. Xin, *SoftGNSS*, Retrieved Apr. 11, 2014, from https://github.com/TMBOC/SoftGNSS

[8] M. Z. H. Bhuiyan, *FGI-GSRx Open Source multi-GNSS Software Receiver*, Retrieved Apr. 11, 2014, from https://github.com/nlsfi/FGI-GSRx

[9] D. Akos, University of Colorado Boulder, *OS GNSS SDR Collection*(2022), Retrieved Apr. 11, 2014, from https://github.com/gnsscusdr/CU-SDR-Collection

[10] S. Yoo, J. D. Yoo, and S Y. Kim, *OS-GNSS-L1CA-SDR-CMEX-EDIT*(2024), Apr. 11, 2014, from https://github.com/abcbank/OS-GNSS-L1CA-SDR-CMEX-EDIT

유 승 수 (Seungsoo Yoo)

2003년 2월 : 건국대학교 전자정보통신공학부 졸업
2005년 2월 : 동 대학원 공학석사
2010년 8월 : 동 대학원 공학박사
2010년 9월 2011년 2월 : 건국대학교 2단계 BK21 위성항법시스템 수신기 기술 연구팀 박사후연구원
2011년 3월~현재 : 건국대학교 전기전자공학부 조교수
<관심분야> GNSS 신호처리, 기계학습 기반 GNSS 재밍 신호 검출 및 추정 등
[ORCID:0000-0002-8648-1540]

유 재 덕 (Jae Duk Yoo)

2024년 2월 : 건국대학교 전자공학과 졸업
2024년 3월~현재 : 건국대학교 전자정보통신공학과 석사과정
<관심분야> GNSS SDR, GNSS 신호처리
[ORCID:0009-0002-3095-2224]

김 선 용 (Sun Yong Kim)

1990년 2월 : 한국과학기술원 전기 및 전자공학과 졸업(최우등)
1993년 2월 : 동 대학원 공학석사
1995년 8월 : 동 대학원 공학박사
1996년~2001년 : 한림대학교 정보통신공학부 조교수
2001년~현재 : 건국대학교 전기전자공학부 교수
<관심분야> 통계학적 신호처리, 이동통신시스템, 통신이론, GPS 항재밍 신호처리 등
[ORCID:0000-0002-4192-2146]