JKSCI

한국컴퓨터정보학회논문지
Journal of The Korea Society of Computer and Information
Vol. 30 No. 11, pp. 1-10, November 2025
https://doi.org/10.9708/jksci.2025.30.11.001

Multiprocessing-Based Parallel Pipeline for Real-Time Multi-Target Tracking on Embedded Systems

Dong-Hyun Kim*, Du-Hwan Hur**, Jeong-Hun Ha*, SeungHwan Bae***,
Kang-Won Seo***, Seung-Hwan Bae****

*M. S. candidate, Vision & Learning Lab, Inha University, Incheon, Korea
**Ph.D. candidate, Vision & Learning Lab, Inha University, Incheon, Korea
***Researcher, Performance analysis Team, LIGNEX1, Pangyo, Korea
****Associate Professor, Vision & Learning Lab, Dept. of Electrical and Computer Engineering, Inha University, Incheon, Korea

[Abstract]

In this paper, we propose a multiprocessing-based parallel pipeline architecture for robust real-time implementation of multi-target designation and tracking algorithms in embedded environments. While the demand for on-site complex AI vision is surging in advanced industries, real-time processing on resource-constrained embedded systems remains a significant technical challenge. The proposed architecture addresses this by separating pre-processing, inference, and post-processing into independent processes, enabling frame-level parallelism. Furthermore, it efficiently distributes computations across CPU, GPU, and NPU resources to minimize idle time and improve overall throughput. On the Jetson AGX Orin platform, the architecture achieves an 88% improvement in FPS compared to a single-process baseline with stable enhancements in tracking accuracy. In contrast, on a server using the TensorRT runtime, performance gains were limited due to conflicts with its internal optimizations. These findings demonstrate that our architecture enables the on-device implementation of complex vision AI models without high-performance servers, making it a key technology for applications such as precision-guided weapons and autonomous drones.

▸ **Key words**: Multi-target Tracking, Embedded System, Multiprocessing, Real-time Processing, Parallel Execution

• First Author: Dong-Hyun Kim, Corresponding Author: Seung-Hwan Bae
*Dong-Hyun Kim (ddongpal0730@inha.edu), Vision & Learning Lab, Inha University
**Du-Hwan Hur (gjenghks@inha.edu), Vision & Learning Lab, Inha University
*Jeong-Hun Ha (jhha@inha.edu), Vision & Learning Lab, Inha University
***SeungHwan Bae (seunghwan.bae@lignex1.com), Performance analysis Team, LIGNEX1
***Kang-Won Seo (kangwon.seo@lignex1.com), Performance analysis Team, LIGNEX1
****Seung-Hwan Bae (shbae@inha.ac.kr), Vision & Learning Lab, Dept. of Electrical and Computer Engineering, Inha University
• Received: 2025. 09. 04, Revised: 2025. 09. 28, Accepted: 2025. 10. 28.

http://www.ksci.re.kr pISSN:1598-849X | eISSN:2383-9945

[요 약]

본 논문은 다층 표적 지정 및 추적 알고리즘을 임베디드 환경에서 실시간으로 안정적으로 수행하기 위한 멀티프로세싱 기반 병렬 파이프라인 구조를 제안한다. 최근 첨단 산업 분야에서 복합적인 인공지능 비전 기술의 현장 적용 요구가 급증하고 있으나, 자원이 제한된 소형 임베디드 시스템에서 이를 실시간으로 처리하는 것은 여전히 기술적 난제로 남아있다. 이에 본 연구는 전처리, 추론, 후처리 단계를 독립된 프로세스로 분리하고, 프로세스 간 큐(queue)를 활용해 프레임 단위의 병렬 처리를 통해 해결하고자 한다. 제안된 구조는 각 연산 단계의 특성에 따라 CPU, GPU, NPU 자원을 효율적으로 분산 배치함으로써 연산 자원의 유휴 시간을 줄이고 전체 처리율을 향상시켰으며, Jetson AGX Orin 및 RTX 3090 서버 환경에서의 성능을 실험적으로 검증하였다. Jetson 환경에서는 FPS가 약 88% 향상되는 안정적인 개선을 보인 반면, 서버의 TensorRT 환경에서는 내부 최적화와의 충돌로 성능 향상이 제한적이었다. 종합적으로 본 연구 결과는 멀티모델 기반 비전 알고리즘의 실시간 적용 가능성을 실험적으로 입증하여 고성능 서버 없이도 복잡한 비전 AI 모델의 온디바이스 구현을 가능하게 하며, 정밀 유도무기, 자율주행 드론 등 다양한 분야에 핵심 기술로 활용될 것으로 기대된다.

▸ **주제어**: 다층 표적 추적, 임베디드 시스템, 멀티프로세싱, 실시간 처리, 병렬 실행

I. Introduction

유도무기와 같은 장거리 정밀타격 시스템에서 안정적인 표적 추적 기술은 시스템의 성능과 직결된다. 특히 약 3km 이상의 거리에서 표적이 영상 내에서 매우 작게 나타나기 때문에, 기존 유도무기 시스템에서는 주표적의 손실이나 추적 실패가 빈번하게 발생하며, 이에 따라 유효 추적 거리를 5km 이상으로 확장하려는 기술적 수요가 증가하고 있다[1].

이를 해결하기 위해 본 연구에서는 다층 표적지정 및 추적 알고리즘을 통해 주표적 추적이 불안정할 경우 주변의 객체(건물 등)를 부표적으로 추적하는 중간 단계 구조를 활용한다. 이 구조는 일시적으로 추적 불가 상황에서도 추적을 이어가도록 하여 유도 중단을 방지하며, 동시에 주표적을 재검출하는 복원 메커니즘을 포함한다. 하지만 이러한 다중 모델 기반 영상처리 알고리즘을 유도무기의 탑재 컴퓨팅 보드에서 실시간으로 처리하기 위해서는 제한된 하드웨어 자원(CPU/NPU/GPU)을 효율적으로 활용하는 최적화가 필요하다.

특히 검출, 추적 등 서로 다른 기능의 모델들을 함께 사용하는 경우, 자원 충돌, 병목, 비동기 처리 지연 등으로 인해 전체 처리 성능이 크게 저하될 수 있다[2]. 기존의 임베디드 영상처리 시스템은 일반적으로 하나의 프로세스 내에서 프레임 획득, 전처리, 추론, 후처리, 결과 출력까지 모든 단계를 순차적으로 처리하는 구조를 따른다. 이러한 단일 프로세스 기반의 순차 처리 방식은 구조가 단순하고 구현이 용이하다는 장점이 있지만, 실시간성과 처리 효율 측면에서는 본질적인 한계를 지닌다[3]. 우선, Figure 1에 묘사된 것처럼, 각 처리 단계가 직렬적으로 연결되어 있기 때문에, 어느 하나의 단계에서 연산 지연이 발생할 경우 이후 단계가 전부 대기 상태에 빠진다. 특히 추론 단계에 복잡한 모델이 투입되는 경우, 전처리나 후처리 단계는 추론이 끝날 때까지 자원을 활용하지 못한 채 유휴 상태로 남게 된다. 이로 인해 전체 시스템 처리 속도가 감소하고, 프레임 지연이 누적되면 실시간 영상 처리에서의 성능 저하로 이어진다[4].

또한, 이러한 구조는 멀티코어 CPU, NPU, GPU와 같은 병렬 연산 자원을 충분히 활용하지 못한다는 한계가 있다. 하드웨어적으로 병렬 처리를 지원하더라도, 소프트웨어 구조가 단일 프로세스 기반이라면 해당 자원은 실제 사용되지 않고 낭비되기 쉽다. 임베디드 시스템처럼 연산 자원이 제한된 환경에서는 이러한 자원 비효율이 더욱 치명적으로 작용한다[5]. 이러한 병목과 자원 낭비는 실시간 영상을 처리하는 과정에서 프레임 손실이나 불규칙한 처리 간격으로 나타나며, 이는 곧 추적 정확도 저하 및 표적 손실로 이어진다[6]. 특히 장거리에서의 안정적인 표적 추적이 요구되는 유도무기 시스템의 경우, 이러한 문제는 전체 시스템의 신뢰도에 직접적인 영향을 미칠 수 있다.

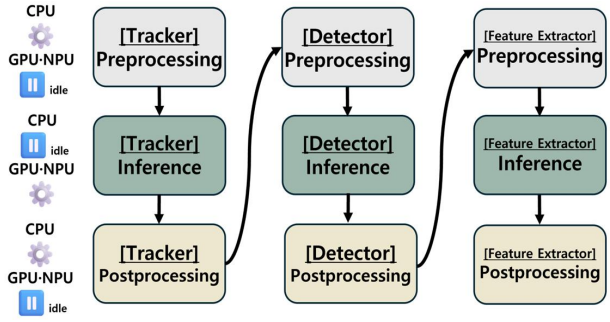

Fig. 1. Heterogeneous resource allocation for the multi-model sequential pipeline.

이러한 순차 처리 방식의 한계를 극복하기 위해 멀티스레딩이나 파이프라인을 활용한 기존 병렬 처리 연구들이 다수 존재하지만 다음과 같은 명확한 한계를 보인다. 대부분 단일 모델의 속도 향상에만 집중하여 여러 종류의 딥러닝 모델이 동시 실행될 때 발생하는 복합적인 자원 충돌 및 데이터 병목 현상을 효과적으로 해결하지 못하는 명확한 한계를 보인다. 또한, 최신 임베디드 보드의 핵심인 NPU까지 포함한 CPU-GPU-NPU 이기종 자원을 연산 특성에 맞춰 효율적으로 할당하는 통합적인 아키텍처에 대한 연구는 초기 단계에 머물러 있다.

본 연구에서는 기존 병렬화 연구들의 한계를 극복하기 위해 멀티프로세싱 기반의 병렬 파이프라인 구조를 제안한다. 제안하는 구조는 단순한 처리 단계 분할을 넘어서, 검출(Detection), 추적(Tracking), 특징맵 추출기(Feature Extractor) 등 주요 컴포넌트를 각각 독립된 프로세스로 분리하여, 연산 특성과 자원 요구에 맞춰 최적화된 실행 환경을 제공하도록 설계되었다. 각 프로세스는 고유한 기능을 수행하며, 프로세스 간에 큐(queue)를 두어 스트림 데이터를 비동기적으로 전달받는다. 이로써 마치 자동차 조립라인처럼 여러 프레임이 파이프라인 상의 각 단계에서 동시에 처리될 수 있어, 처리량(throughput)은 높이고 지연 시간(latency)은 감소하게 된다[4].

또한 각 연산 블록의 특성에 따라 전처리와 후처리는 CPU에서, 추론은 NPU와 GPU 에서 수행되도록 분리함으로써, 연산 자원의 대기 시간을 최소화하고 전체 자원 활용률을 극대화하였다[7]. 이러한 병렬 파이프라인 설계는 단일 프로세스 기반 구조에서 발생하는 병목 문제를 해소하고, 실시간 환경에서도 검출-추적-특징맵 추출이 안정적으로 동시에 수행될 수 있는 비전 처리 시스템의 구현을 가능하게 한다.

본 논문에서는 제안하는 구조의 설계 원칙과 구현 방식에 대해 자세히 기술하고, 실제 임베디드 환경에서 수행한 실험을 통해 프레임 처리 속도, 자원 활용도, 추적 정확도 등의 정량적 지표를 기반으로 성능 향상 효과를 입증한다.

II. Related Works

본 장에서는 임베디드 비전 시스템에서 실시간 처리를 달성하기 위한 멀티프로세싱 구조 설계와 파이프라인 최적화 기법, 그리고 다층 표적 추적과 외형 기반 일관성 평가에 관련된 기존 연구들을 소개한다. 또한, 본 연구에서 제안하는 병렬 파이프라인 구조가 기존 연구들과 비교해 가지는 차별점에 대해서도 서술한다.

1. Embedded Vision System Optimization

임베디드 영상 처리 분야에서는 실시간 추론 성능을 달성하기 위한 다양한 접근이 연구되어 왔다. 대표적으로 FPGA, GPU, NPU 등 전용 하드웨어 가속기를 활용한 병렬화와 더불어, 모델 경량화 및 연산 병목 제거를 위한 소프트웨어 최적화 기법들이 제안되어 왔다[8]. 최근에는 CPU와 AI 가속기를 결합한 이기종(heterogeneous) 시스템에서 파이프라인 최적화를 통해 성능을 극대화하는 방안이 특히 주목받고 있다. CPU 전처리와 NPU 추론을 병렬로 수행하기 위한 이중 버퍼링(double buffering) 기법은 NPU의 유휴 대기 시간을 줄여 FPS를 효과적으로 향상시키는 성과를 보였다. 그러나 이러한 접근은 단일 모델의 순차적 처리 속도를 높이는 데 국한되며, 본 연구처럼 검출, 추적 등 여러 모델이 동시에 자원을 요구하는 복합 시스템의 병목 현상은 해결하지 못한다는 명확한 한계가 있다. 이처럼 입출력 단계와 추론 단계를 겹쳐서 수행하는 방법은 메모리 사용이 소폭 증가하는 trade-off 관계가 있으나, 실시간성이 중요한 임베디드 환경에서의 추론 속도 향상과 연산 자원 활용률 증가라는 측면에서 실질적인 성능 개선을 달성한다.

또한, 멀티스레드와 버퍼 큐를 통해 병목을 완화하는 방식도 제안되었다[7]. 하지만 이 역시 단일 기능의 처리 흐름을 최적화하는 데 중점을 둔 것으로, 서로 다른 연산 특성을 가진 복합 모델들을 동시에 운용할 때 발생하는 자원 충돌 문제에 대한 해결책을 제시하지는 못했다. 이러한 선행 연구들은 모두 프레임 간 지연 시간(latency)을 줄이면서도 처리량을 유지할 수 있는 장점을 가지며, 네트워크 지연이나 연산 대기 상황에서도 연산 흐름의 안정성을 확보할 수 있다.

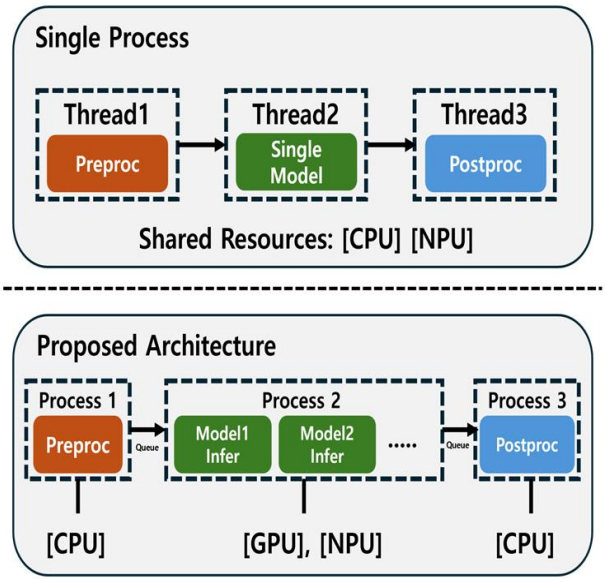

Fig. 2. Comparison of pipeline architectures with prior art.

이와 같은 연구들은 공통적으로 병렬 파이프라인을 활용한 연산 자원 최적화가 임베디드 환경에서 실시간 성능 확보의 핵심임을 보여준다. 그러나 대부분의 연구는 단일 기능 또는 단일 모델 기반 처리 구조에 집중되어 있으며, 검출, 추적, 특징맵 추출과 같은 복합적인 모델을 동시에 운영하는 구조에서의 연산 분산 및 프로세스 병렬화에 대한 연구는 상대적으로 부족한 실정이다.

2. Multiprocessing-Based Vision Pipeline

영상 처리 시스템에서 프레임 처리 지연을 줄이고 전체 처리량을 향상시키기 위해, 멀티프로세싱 기반의 병렬 파이프라인 구조가 다양한 응용 분야에서 적용되고 있다[9]. 이 구조에서는 영상의 전처리, 추론, 후처리 단계를 서로 독립적인 프로세스 혹은 스레드로 분리하고, 각 단계 간에 FIFO 큐(First-In-First-Out queue)를 배치하여 프레임 단위로 데이터를 전달하는 방식이 일반적으로 사용된다[10]. 이와 같은 구조는 각 단계가 병렬로 작동하여 다수의 프레임이 동시에 서로 다른 처리 단계에 위치할 수 있게 하며, 결과적으로 시스템의 전체 처리량(throughput)을 증가시킨다. 이러한 멀티프로세싱 구조는 GStreamer, ROS 등에서 널리 채택되어 왔다[4, 10]. 그러나 이들 대부분의 구현은 CPU와 GPU 간의 연산 분산에 그쳤을 뿐, 최신 임베디드 보드의 핵심인 NPU를 파이프라인에 유기적으로 통합하여 CPU-GPU-NPU 자원 전체를 동적으로 관리하는 포괄적인 아키텍처는 제시하지 못했다.

또한, 프레임 단위로 동작하는 병렬 처리 구조는 연산 병목이나 프레임 유실로 인해 발생할 수 있는 실시간성 저하

Table 1. Comparative analysis

Feature	Prior study[7]	Proposed
Parallelism	Multi-thread	Multi-process
Task	Single Model	Multi-model
Resources	CPU, NPU	CPU, NPU, GPU
Contribution	Race Condition	Stability Optimization

문제를 완화하는 데 효과적이다. 처리 지연이 누적되더라도, 독립된 큐 구조를 통해 이후 프레임의 흐름이 완전히 차단되지 않으며, 연산 대기 중에도 다른 단계는 병렬적으로 진행될 수 있기 때문이다[3]. 그러나 대부분의 기존 연구는 단일 비전 모델의 파이프라인 처리 구조에 중점을 두고 있으며, 검출기, 추적기, 특징맵 추출기 등 서로 다른 연산 특성을 갖는 복합 딥러닝 모듈이 결합된 시스템에서의 멀티프로세싱 구조 설계는 거의 다루어지지 않았다. 특히, 이기종 연산 자원(CPU, NPU 와 GPU)을 활용하면서도 모델 간 병목과 자원 경합을 최소화할 수 있는 구조적 설계에 대한 연구는 아직 충분히 이루어지지 않았다. 앞서 분석한 기존 연구들과 본 논문이 제안하는 구조의 핵심적인 차이점은 Figure2와 Table 1에 요약되어 있다. 기존 연구들은 단일 모델 처리나 제한적인 자원 활용에 머무른 반면, 제안된 구조는 복합 다중 모델 환경에서 이기종 자원 전체를 효율적으로 활용하는 데 중점을 둔다. 이러한 CPU-NPU-GPU 자원 분산 전략을 통해 실시간 추적 성능을 극대화하기 위한 차별화된 구조의 설계 원칙과 각 모듈의 구체적인 동작 방식에 대해서는 다음 장에서 상세히 기술한다.

III. Methodology

본 장에서는 임베디드 환경에서 다층 표적 추적의 실시간 성능을 향상시키기 위한 구조를 검토하고, 이를 위해 제안한 멀티프로세싱 기반 병렬 파이프라인 구조의 설계 및 구현 방법을 소개한다. 또한, 제안 구조의 효율적인 실행을 위한 프로세스 분할 방식과 연산 자원 활용 전략에 대해서도 함께 설명한다.

1. Framework Overview

본 절에서는 제안하는 멀티프로세싱 기반 병렬 파이프라인 구조의 전반적인 동작 흐름과 프레임 처리 방식에 대해 설명한다. 시스템은 전처리(Preprocessing), 추론(Inference), 후처리(Postprocessing)의 세 단계로 구성되며, 각 단계는 독립된 프로세스로 정의되고, 프로세스

간 통신에는 FIFO 구조인 큐(queue)가 배치되어 프레임 데이터를 순차적으로 전달한다. 이 구조의 전체 흐름과 병렬 처리 방식은 Figure 3에 시각적으로 정리되어 있다. 프레임 단위로 분리된 데이터는 각 프로세스를 통과하면서 파이프라인 구조상 병렬로 처리된다. 예를 들어 시간 step t 에서, 전처리 프로세스는 프레임 $N + 1$, 추론 프로세스는 프레임 N 후처리 프로세스는 프레임 $N - 1$을 각각 동시에 처리한다. 이와 같은 구조는 frame-level pipeline parallelism을 실현하며, 전체 처리율(FPS)을 향상시키고, 시스템 병목 현상을 완화한다. 또한, 이 구조는 각 프로세스가 독립적으로 동작하므로 특정 단계에서 일시적인 지연이 발생해도 시스템 전체가 정지하지 않으며, 큐를 활용한 흐름 제어를 통해 프레임 손실이나 과부하를 방지할 수 있다. 이러한 특성은 실시간 처리를 요구하는 임베디드 비전 시스템에서 안정성과 신뢰성을 확보하는 데 중요한 요소가 된다. 제안된 병렬 파이프라인 구조의 실제 실행 흐름은 Algorithm 1의 수도코드를 통해 요약할 수 있다. 각 처리 단계는 독립적인 워커 프로세스로 정의되며, FIFO 큐를 통해 프레임 데이터를 비동기적으로 전달받아 병렬 처리를 수행한다.

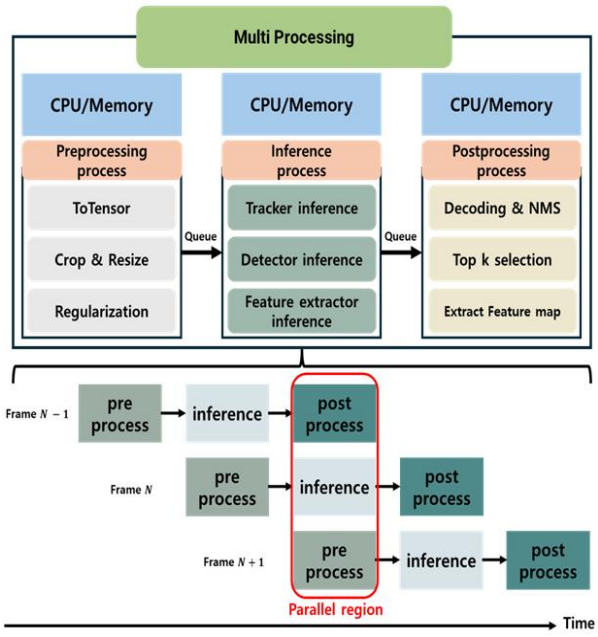

Fig. 3. Overview of the proposed multiprocessing-based parallel pipeline framework.

2. Processing Modules

본 절에서는 제안된 병렬 파이프라인을 구성하는 세 개의 주요 프로세스인 전처리, 추론, 후처리의 역할과 동작 방식에 대해 상세히 설명한다. 각 프로세스는 하나의 파이프라인 단계를 담당하며, 입력 및 출력 큐를 통해 상호 비동기적으로 데이터를 주고받는다. 이 구조는 각기 다른 프레임이 동시에 서로 다른 단계에서 병렬로 처리되는 구조로 구현된다.

2.1 Preprocessing Process

전처리 프로세스는 카메라 또는 영상 센서로부터 실시간으로 프레임을 수신하고, 이후 단계의 추론 모델이 요구하는 입력 형식으로 영상을 변환하는 역할을 수행한다. 이 과정에서는 먼저 프레임의 해상도를 변경하고, 필요 시 관심 영역만 남기도록 크기를 조정하며, 영상의 품질 향상을 위해 간단한 노이즈 제거 필터링을 적용한다. 이어서 픽셀 값을 정규화하고, 데이터를 float32 형태의 텐서로 변환한 뒤, 딥러닝 프레임워크에 맞는 입력 구조(N, C, H, W)로 재배열한다. 마지막으로 배치 차원을 추가하여 추론 프로세스에서 즉시 활용할 수 있도록 준비된 데이터를 입력 큐에 전달한다. 해당 프로세스는 연산량이 비교적 낮아 CPU에서 실행되며[11], 다음 프레임을 미리 준비함으로써 입력 대기 시간을 최소화하고 파이프라인 전체의 효율성을 높이는 데 기여한다. 또한, 이 단계는 항상 최신 프레임을 우선 처리하므로 시스템 지연 누적을 방지하고 실시간성 유지에 중요한 역할을 한다.

2.2 Inference Process

추론 프로세스는 전처리된 프레임을 입력 받아, 딥러닝 기반의 검출기 및 추적기를 실행하는 단계이다. 이 프로세스에서는 YOLOX[12] 계열의 객체 검출기와 SiamTPN[13] 기반의 추적기가 함께 사용되며, 임베디드 시스템의 연산 특성에 맞게 검출기를 NPU에서 추적기를 GPU에서 동작하도록 설계되었다. 프레임이 입력되면 먼저 YOLOX 검출기가 각 표적의 위치를 bounding box 형태로 검출하고, 이어서 SiamTPN 추적기가 해당 위치 정보를 기반으로 추적을 수행한다. 이 과정에서 표적에 대한 분류 확률, 특징 벡터, 좌표 정보 등이 함께 산출된다. 추론 결과는 원본 프레임의 타임스탬프와 함께 패키징되어 출력 큐에 저장되며, 이후 후처리 프로세스에서 이를 활용한다. 이러한 전체 추론 절차는 Algorithm 1의 수도코드에 요약되어 있으며, 각 프레임은 독립적으로 처리되어 병렬 파이프라인 상에서 효율적인 연산 흐름을 유지할 수 있도록 한다.

추론이 수행되는 동안 전처리 프로세스는 다음 프레임을 준비하고, 후처리 프로세스는 이전 프레임을 처리함으로써 세 프로세스가 동시에 작동한다. 이를 통해 CPU, GPU와 NPU의 자원 유휴 시간이 최소화되며, 연산 자원

Algorithm 1 Multiprocessing-based Parallel Processing

```
Initialize queues: Q_pre, Q_infer, Q_post
Launch three parallel processes:
function PreprocessingProcess
    for each frame index t = 0, 1, 2, ... do
        F_t ← ReadNextFrame(t)
        P_t ← Preprocess(F_t)
        Q_pre.put(P_t, t)
    end for
end function
function InferenceProcess
    while True do
        Peek next item in Q_pre to get t
        if t − 1 ≥ 0 then
            (P_{t−1}, t) ← Q_pre.get()
            I_{t−1} ← Inference(P_{t−1})
            Q_infer.put(I_{t−1}, t)
        else
            Skip and continue
        end if
    end while
end function
function PostprocessingProcess
    while True do
        Peek next item in Q_infer to get t
        if t − 2 ≥ 0 then
            (I_{t−2}, t) ← Q_infer.get()
            R_{t−2} ← Postprocess(I_{t−2})
            Q_post.put(R_{t−2}, t)
        else
            Skip and continue
        end if
    end while
end function
```

Algorithm 1. Multiprocessing-based parallel processing pseudo-code

활용도가 극대화된다. 특히 이 단계는 파이프라인의 병목이 발생할 수 있는 구간이므로, 추론 처리 속도와 정확도의 균형을 고려하여 모델 경량화 및 입력 해상도 조정 등의 최적화 기법이 적용된다.

2.3 Postprocessing Process

후처리 프로세스는 추론 결과를 바탕으로 주표적 및 부표적의 추적 상태를 관리하고, 최종적인 결과 출력 또는 시스템 제어 신호로 변환하는 역할을 수행한다. 가장 핵심적인 기능은 다층 표적 지정 및 전환 알고리즘의 실행으로, 추론 결과를 기반으로 추적 신뢰도 점수를 평가하여 주표적의 추적 가능 여부를 판단한 후, 주표적이 추적 가능한 경우 부표적에서 주표적으로의 표적 전환을 수행한다. 또한, 추적 신뢰도 점수 평가의 안정성 및 일관성을 높이기 위해 Kalman filter[14] 기반의 시계열 예측 모델을 적용하여 위치 노이즈를 보정하고, 화면 흔들림을 완화한다.

최종적으로 박스 정보와 조준점을 영상 위에 시각적으로 출력하거나, 외부 시스템으로 유도 명령 형태의 데이터를 전송한다. 이 프로세스는 CPU 상에서 실행되며, 추론 단계와 병렬로 처리됨으로써 파이프라인 병렬성이 완전히 유지된다.

3. Execution Control & Resource Allocation

본 절에서는 제안된 멀티프로세싱 병렬 파이프라인 구조의 안정적인 실행을 보장하고, 연산 자원의 활용도를 극대화하기 위한 제어 전략과 자원 분배 방식을 설명한다. 특히, 프레임 간 처리 지연이나 자원 경합이 전체 시스템 신뢰도에 영향을 미칠 수 있는 임베디드 환경에서는, 각 프로세스의 연산 흐름을 정밀하게 조율하고 연산 자원을 효율적으로 배분하는 것이 필수적이다. 제안 구조는 각 처리 단계 간에 FIFO 구조인 큐(queue)를 배치하여, 프레임 단위의 비동기 처리를 실현한다. 각 큐는 다음 프로세스가 데이터를 읽어가는 순간 이전 데이터를 제거하는 선입선출 방식으로 동작하며, 프레임의 처리 순서를 보장하면서도 각 단계의 연산 속도 차이를 효과적으로 흡수하는 완충(buffering) 역할을 수행한다. 이를 통해 특정 프로세스에서 일시적인 지연이 발생해도 전체 파이프라인이 정지하지 않고 안정적으로 동작할 수 있다.

프로세스는 전처리-추론-후처리로 구분되며, 각각 독립적인 프로세스로 실행된다. 전처리와 후처리는 연산량이 낮고 주기적인 작업이므로 CPU 상에서 실행되며, 고연산량의 추론 단계는 GPU와 NPU에서 수행되도록 분리되어 있다. 이와 같은 연산 특성 기반의 자원 분산은 CPU, GPU와 NPU의 유휴 시간을 최소화하고, 전체 시스템 처리율을 향상하는 데 기여한다.

추론 단계에서는 복수의 딥러닝 모델(예: 객체 검출기, 추적기 등)이 비동기적으로 실행되며, 각 모델의 출력은 공유 메모리 영역에 저장된다. 모든 모델의 결과가 수신되면 이를 통합하여 후속 단계에 전달하는 방식으로 설계되어, 동기화 지연을 줄이면서도 안정적인 연산 흐름을 유지할 수 있다.

이러한 실행 제어 및 자원 분배 전략은 제안된 병렬 파이프라인 구조의 실시간 처리 성능을 뒷받침하는 핵심 요소로 작용하며, 제한된 연산 자원 환경에서도 안정적인 다층 표적 추적을 가능하게 한다.

IV. Experiments

본 장에서는 제안한 멀티프로세싱 기반 병렬 파이프라인 구조의 성능을 정량적으로 검증하기 위해 수행한 실험 환

경과 결과를 소개한다. 실험은 고성능 서버 환경(On-GPU)과 임베디드 환경(On-Board)에서 각각 수행되었으며, 단일 프로세스(SP) 구조와 멀티프로세싱(MP) 구조를 동일한 조건 하에서 비교하였다. 또한, 추적 성능을 파악하기 위해 11가지 지표를 기준으로 평가를 수행하였다.

1. Experimental Setup

본 절에서는 제안한 멀티프로세싱 기반 병렬 파이프라인 구조의 성능을 검증하기 위한 실험 환경과 평가 지표에 대해 설명한다. 실험은 다른 하드웨어환경에서 수행되었으며, 각각 서버 환경(On-GPU)과 임베디드 환경(On-Board)으로 구성된다. Table 2은 두 실험 환경의 주요 하드웨어 사양을 정리한 것이다. 또한, 제안한 구조의 성능을 단일 프로세스 구조와 비교하여 정량적으로 평가하였다.

Table 2. Hardware Specification

	Type	Hardware Specification
On-GPU	CPU	Intel Xeon Gold 6242 2ea
	RAM	DDR4 ECC REG 32 GB 8ea (256GB)
	GPU	NVIDIA RTX 3090 24GB
On-Board	CPU	ARM Cortex A78AE 2.2GHz
	RAM	LPDDR5 64GB
	GPU	NVIDIA Ampere 기반 Jetson AGX Orin 2048 CUDA Cores, 64 Tensor Cores
	NPU	NVDLA v2

또한, 성능 평가는 유도무기 운용 환경을 시뮬레이션한 14-bit 합성 적외선 영상 데이터셋을 기반으로 수행되었다. 데이터셋은 총 6종의 표적(3종 전차, 3종 차량)을 포함하며, 다양한 거리(1.5–5.0 km), 계절(봄·여름·겨울), 시간대(09시, 12시, 19시, 22시) 조건을 반영한다. 본 연구에서는 이 데이터셋의 테스트 시퀀스 200개를 활용하여 주표적과 부표적이 혼합된 다양한 상황에서 실험을 진행하였고, 각 시나리오별 평균값을 기준으로 11개 정량 지표를 평가하였다. 결과는 대부분의 지표에서 일관된 향상 경향을 보여, 제안된 구조의 안정성과 신뢰성을 입증하였다.

사용된 11개의 성능 평가지표는 다음과 같다:

- SUB Acquisition Range Mean / Std: 부표적의 최초 포착 거리의 평균 및 표준편차를 의미하며, 부표적 획득 성능의 일관성을 평가한다.
- Switching Success Rate: 부표적에서 주표적으로 전환 후 10프레임 동안 IoU ≥ 0.5를 유지한 비율로 정의되며, 주표적 전환의 안정성을 나타낸다.
- Main Acquisition Range Mean / Std: 주표적 최초 포착 거리의 평균 및 표준편차로, 주표적 검출 능력을 나타낸다.
- Main RMSE: 중심 좌표 기준의 L2 오차를 측정하며, 픽셀 단위 추적 정확도를 나타낸다.
- Main Error Mean / Std: 위 오차를 실제 거리 단위(meter)로 변환한 평균 및 분산 값이다.
- Main Tracking Success Rate: 추적 성공률 지표로, IoU가 0.1 이하로 떨어지는 실패가 3회 이상 발생하지 않으면 성공으로 간주한다.
- Main ±1m Accuracy Rate: GT 기준과의 중심 오차 평균이 특정 거리 구간(400~150m)에서 ±1m 이하인 프레임 비율로, 정밀 추적 성능을 나타낸다.
- Main Stability Rate: 동일 구간 내 중심 오차의 최대-최소 차이가 1m 이내인 비율로, 추적 안정성을 나타낸다.

이러한 지표들은 부표적 추적과 주표적 전환의 초기 단계부터, 전체 추적 구간에서의 정밀도와 일관성까지 종합적으로 평가할 수 있도록 구성되어 있다.

2. Experimental Results

본 절에서는 제안한 멀티프로세싱 기반 병렬 파이프라인 구조(MP)와 기존 단일 프로세스 기반 구조(SP)를 비교하여 성능 향상을 정량적으로 분석한다. 비교는 서버(RTX 3090 기반 GPU)와 임베디드 보드(Jetson AGX Orin 기반 NPU) 각각에서 수행되었으며, 총 200개 시나리오에 대해 평가되었다. 분석에는 총 11개의 정량적 지표가 사용되었으며, 이들은 주/부 표적의 포착 거리, 중심 좌표 오차, 표적 추적 성공률, 정확도, 안정성 등을 포함한다.

Table 3는 각 실험 환경에서의 SP 및 MP 구조별 성능 수치를 정리한 결과이다. 멀티프로세싱(MP) 구조의 효과는 하드웨어 플랫폼과 런타임에 따라 상이하게 나타났다. 먼저 서버 환경(RTX 3090)에서 단일 프로세스 기준 성능을 보면 TensorRT는 CUDA 대비 더 높은 FPS(77.07 vs. 64.92)를 기록하였다. 이는 TensorRT가 커널 퓨전, 레이어 병합, 메모리 재사용 등 단일 실행 환경에 특화된 최적화를 적용했기 때문이다. 그러나 MP 구조를 적용하면 양상이 달라진다. CUDA의 경우 단일 프로세스 기준 성능에서 최적화가 단순하여, 전처리·추론·후처리의 병렬 실행을 통해 자원 활용률이 개선되면서 FPS가 소폭 향상되었다(64.92→65.77, +1.3%). 반면 TensorRT는 단일 프로세스 기준 성능에서 이미 최적화가 극대화된 상태였기 때문에,

Table 3. Performance comparison of single-process (SP) and multiprocessing (MP) structures on GPU (RTX 3090) and embedded (Jetson Orin) environments across multiple evaluation metrics.

	On-GPU (RTX 3090)				On-Board (Jetson Orin)	
	CUDA		TRT (FP32)		TRT (FP32)	
	SP	MP	SP	MP	SP	MP
Sub Acquisition Range Mean (m)	3543.25	3543.25	3543.25	3543.25	3543.25	3543.25
Sub Acquisition Range Std (m)	888.88	888.88	888.88	888.88	888.88	888.88
Switching Success Rate (%)	0.85	0.85	0.85	0.86	0.86	0.86
MAIN Acquisition Range Mean (m)	2030.93	2019.02	2031.49	2017.86	2032.03	2020.11
MAIN Acquisition Range Std (m)	334.27	337.79	334.36	336.46	333.82	337.99
Main RMSE (pixel)	8.91	8.17	9.14	8.24	9.33	8.36
Main Error Mean (m)	1.02	1.14	1.00	1.26	1.12	1.18
Main Error Std (m)	1.34	1.62	1.30	1.70	1.50	1.57
Main Tracking Success Rate (%)	0.66	0.76	0.64	0.77	0.63	0.76
Main ±1m Accuracy Rate (%)	0.89	0.91	0.90	0.90	0.90	0.90
Main Stability Rate (%)	0.84	0.83	0.83	0.85	0.85	0.85
Frames Per Second (FPS)	64.92	**65.77**	**77.07**	71.69	25.80	**48.63**

MP 구조 적용 시 전처리와 후처리가 별도 프로세스로 분리되면서 엔드투엔드 최적화 경로가 단절되고 프로세스 간 데이터 전달 지연이 상대적으로 크게 작용하였다. 그 결과 FPS가 오히려 감소하였다(77.07→71.69, -6.9%). 한편 Jetson Orin 임베디드 환경에서는 다른 결과가 관찰되었다. Jetson은 CPU·GPU·NPU가 단일 SoC에 통합된 구조로, 단일 프로세스 기준 성능에서는 각 자원이 순차적으로 대기하는 구간이 많아 효율이 낮다. 그러나 MP 구조에서는 CPU, NPU, GPU가 병렬로 동작하면서 자원 유휴 시간이 크게 줄었고, 이 효과가 TensorRT 실행의 IPC 오버헤드보다 훨씬 크게 작용하였다. 그 결과 FPS가 25.80에서 48.63으로 약 88.4% 향상되었으며, 이는 Jetson 환경에서 MP 구조가 실시간성 확보에 결정적인 이점을 제공함을 보여준다. 또한, 주표적의 중심 오차(Main RMSE)는 9.33에서 8.36으로 감소하였고, 평균 오차(Main Error Mean)와 표준편차(Main Error Std) 역시 안정적으로 유지되었다. 추적 성공률 지표(Main Tracking Success Rate)는 0.63에서 0.76으로 향상되었으며, 주표적 정확도(Main ±1m Accuracy Rate) 또한 모든 구간에서 안정적으로 유지되었다. 이는 MP 구조가 Jetson 환경에서 처리 속도뿐만 아니라 추적 정밀도와 안정성 지표에서도 일관된 개선을 제공함을 의미한다.

종합하면, CUDA와 TensorRT의 단일 프로세스 기준 성능 수준 차이, 그리고 서버 GPU와 Jetson SoC의 하드웨어 구조 차이가 MP 구조의 효과를 결정하였다. 서버 환경에서는 TensorRT가 단일 프로세스 기준에서 이미 최적화가 충분히 이루어져 있어 MP 구조가 오히려 불리하게 작용했으며, Jetson 환경에서는 단일 프로세스 기준의 비효율성이 커서 MP 구조의 병렬화 이점이 크게 나타난 것이다.

V. Conclusions

본 논문에서는 임베디드 환경에서 다층 표적 지정 및 추적 알고리즘을 안정적으로 수행하기 위한 멀티프로세싱 기반 병렬 파이프라인 구조를 제안하였다. 제안된 구조는 전처리, 추론, 후처리 단계를 독립된 프로세스로 분리하고, 프로세스 간 큐(queue)를 통해 데이터를 전달함으로써 프레임 단위의 병렬 처리를 실현한다. 이를 통해 연산 자원의 유휴 시간을 최소화하고, 전체 시스템 처리속도를 향상시키는 효과를 달성하였다. 실험 결과, Jetson AGX Orin 기반 임베디드 환경에서는 FPS가 약 88.4% 향상되는 등 실시간성 측면에서 뚜렷한 개선을 보였으며, 중심 오차, 추적 성공률, 정확도 등 주요 지표에서도 안정적인 성능이 확인되었다. 반면, 서버 환경의 TensorRT 기반 실행에서는 내부 최적화 구조와 멀티프로세싱 간의 충돌, 자원 경합 등의 영향으로 FPS가 오히려 감소하는 현상이 나타났으며, 이는 하드웨어 구조와 런타임 특성에 따라 병렬 구조의 효과가 달라질 수 있음을 시사한다.

종합적으로, 본 연구는 멀티모델 기반의 영상 처리 알고리즘을 임베디드 시스템에 실시간으로 적용할 수 있는 일반화된 구조적 해법을 제시하였으며, CPU-GPU-NPU 간 이기종 자원을 효율적으로 활용한 병렬 처리 구조가 실시간성과 안정성을 동시에 확보할 수 있음을 실험적으로 입증하였다. 이는 모델 중심의 최적화 연구를 시스템 수준의 병렬화 구조로 확장한 것으로, 임베디드 비전 처리의 학문적 기반을 확장하는 의의를 가진다.

또한 제안된 구조는 군사 표적 추적뿐만 아니라 자율주행, 스마트시티 감시, 산업용 검사 등 다양한 실시간 임베디드 비전 응용으로 확장 가능하며, 저전력·저지연 환경에

서도 효율적인 병렬 처리를 지원할 수 있다. 향후 연구에서는 다양한 임베디드 플랫폼에서의 일반성과 확장성을 높이기 위해, 동적 자원 할당, 적응형 큐 관리, 프레임 스케줄링 기법 등을 통합하여 구조를 고도화하고, 플랫폼별 런타임 특성에 따른 최적화 전략을 정립하는 방향으로 나아갈 것이다.

ACKNOWLEDGEMENT

This research is performed based on the cooperation with Inha University-LIG Nex1 Cooperation.

REFERENCES

[1] Behzad Mirzaei, et al., "Small object detection and tracking: a comprehensive review," Sensors, vol. 23, no. 15, pp. 6887, Aug. 2023. DOI: https://doi.org/10.3390/s23156887

[2] Qingbo Ji, et al., "Real-time embedded object detection and tracking system in Zynq SoC," EURASIP Journal on Image and Video Processing, vol. 2021, pp. 1-16, Jan. 2021. DOI: https://doi.org/10.1186/s13640-021-00561-7

[3] Pierre Greisen, et al., "An FPGA-based processing pipeline for high-definition stereo video," EURASIP Journal on Image and Video Processing, vol. 2011, pp. 1-13, Jan. 2011. DOI: https://doi.org/10.1186/1687-5281-2011-18

[4] Abhinav Goel, et al., "Efficient computer vision on edge devices with pipeline-parallel hierarchical neural networks," Proceedings of the 27th Asia and South Pacific Design Automation Conference (ASP-DAC), pp. 189-194, Taipei, Taiwan, Jan. 2022. DOI: https://doi.org/10.1109/ASP-DAC52403.2022.9712493

[5] Sparsh Mittal, "A survey of FPGA-based accelerators for convolutional neural networks," Neural Computing and Applications, vol. 32, no. 4, pp. 1109-1139, Feb. 2020. DOI: https://doi.org/10.1007/s00521-018-3761-1

[6] Yu-Hsin Chen, et al., "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks," IEEE Journal of Solid-State Circuits, vol. 52, no. 1, pp. 127-138, Jan. 2016. DOI: https://doi.org/10.1109/JSSC.2016.2616357

[7] Sehyeon Oh, Yongin Kwon, Jemin Lee, "Optimizing Real-Time Object Detection in a Multi-Neural Processing Unit System," Sensors, vol. 25, no. 5, pp. 1376, Mar. 2025. DOI: https://doi.org/10.3390/s25051376

[8] Murad Qasaimeh, et al., "Comparing energy efficiency of CPU, GPU and FPGA implementations for vision kernels," Proceedings of the IEEE International Conference on Embedded Software and Systems (ICESS), pp. 1-8, Las Vegas, USA, Jun. 2019. DOI: https://doi.org/10.1109/icess.2019.8782524

[9] Yuhao Zhu, et al., "Euphrates: Algorithm-SoC Co-design for Low-Power Mobile Continuous Vision," arXiv preprint, arXiv:1803.11232, Mar. 2018. DOI: https://arxiv.org/abs/1803.11232

[10] Yu-Ping Wang, et al., "Tzc: Efficient inter-process communication for robotics middleware with partial serialization," Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 7805-7812, Macau, China, Nov. 2019. DOI: https://doi.org/10.1109/iros40897.2019.8968462

[11] Xubin Wang, Weijia Jia, "Optimizing Edge AI: A Comprehensive Survey on Data, Model, and System Strategies," arXiv preprint, arXiv:2501.03265, Jan. 2025. DOI: https://doi.org/10.48550/arXiv.2501.03265

[12] Ge Zheng, Liu Songtao, Feng Wang, Zeming Li, Jian Sun, "YOLOX: Exceeding YOLO Series in 2021," arXiv preprint, arXiv:2107.08430, Jul. 2021. DOI: https://doi.org/10.48550/arXiv.2107.08430

[13] Daitao Xing, et al., "Siamese transformer pyramid networks for real-time UAV tracking," Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV), pp. 2449-2458, Waikoloa, USA, Jan. 2022. DOI: https://doi.org/10.1109/wacv51458.2022.00196

[14] Rudolph Emil Kalman, "A new approach to linear filtering and prediction problems," Journal of Basic Engineering, vol. 82, no. 1, pp. 35-45, Mar. 1960. DOI: https://doi.org/10.1115/1.3662552

Authors

Dong-Hyun Kim received the B.S. degree in Computer Science and Engineering from Inha University in 2025, and is currently pursuing an M.S. - Ph.D. integrated course degree with the Department of Electrical and Computer Engineering at Inha University. His current research interests are object detection, efficient learning, and model compression.

Du-Hwan Hur received the B.S. degree in Computer Engineering from Hanbat National University in 2021, and is currently pursuing the M.S. - Ph.D. integrated course degree with the Department of Electrical Computer Engineering at Inha University, Korea. His current research interests are object detection, model compression, efficient learning, and on-device AI.

Jeong-Hun Ha received the B.S. degree in Information and Communication Engineering from Inha University in 2025. He is currently pursuing an M.S. - Ph.D. integrated course degree in artificial intelligence with the Department of Electrical Computer Engineering at Inha University. His current research interests include visual object tracking and model compression.

SeungHwan Bae received the B.S and M.S degrees in Medical IT Engineering and Future Convergence Technology from Soonchunhyang University. Korea, in 2019 and 2021, respectively.

SeungHwan Bae joined the Senior Researcher of the Performance analysis Team at LIGNEX1, Pangyo, Korea, in 2022. He is currently a Senior Researcher in the Performance Analysis Team, LIGNEX1. He is interested in AI and Image Processing.

Kang-Won Seo received the B.S. and M.S. degrees in Electrical and Electronics Engineering from Chung-Ang University, Seoul, Republic of Korea, in 2021 and 2023, respectively.

He is currently with LIG Nex1 Co., Ltd., Seongnam, Republic of Korea. His research interests include computer vision, deep learning, and embedded AI systems.

Seung-Hwan Bae received the BS degree in information and communication engineering from Chungbuk National University, in 2009 and the MS and PhD degrees in information and communications from the Gwangju Institute of Science and Technology (GIST), in 2010 and 2015, respectively. He was a senior researcher at Electronics and Telecommunications Research Institute (ETRI) in Korea from 2015 to 2017. He was an assistant professor in the Department of Computer Science and Engineering at Incheon National University, Korea from 2017 to 2020. He is currently an Associate Professor with the Department of Computer Engineering at Inha University, His research interests include object tracking, object detection, generative model learning, continual learning, on-device ML, etc.