Transactions of KSAE, Vol. 34, No. 7, pp.739-749 (July, 2026)

pISSN 1225-6382 / eISSN 2234-0149
DOI http://dx.doi.org/10.7467/KSAE.2026.34.7.739

저비용 웹캠 기반 듀얼 카메라의 프레임 교차를 이용한 FPS 향상

박 범 수 · 정 호 진*

한국교통대학교 자동차공학과

FPS Enhancement using Frame Crossings of Low Cost Webcam-Based Dual Cameras

Beomsu Park · Hojin Jung*

Department of Automotive Engineering, Korea National University of Transportation, Chungbuk 27469, Korea
(Received 14 November 2025 / Revised 25 February 2026 / Accepted 26 February 2026)

Abstract : This study proposed a method to enhance FPS using dual cameras. The role of cameras in autonomous driving technology is becoming increasingly important. However, while high-performance cameras with high FPS can improve autonomous driving performance, they also increase costs. Thus, to solve this problem, we presented a method that replaces one high-cost, high-FPS camera with two low-cost, low-FPS cameras. This approach intentionally asynchronizes the operation of the two camera sensors when using a processor operating at the same cycle as the high FPS. To verify that the images from the two cameras were indeed asynchronized, the timer operation time displayed in the captured camera image files and the time when the corresponding image was saved were cross-validated. By considering real-time applications, this algorithm was verified in a desktop PC and a mini PC environment. The results confirmed that two 30 FPS cameras effectively generate near 60 FPS data through an asynchronous combination of FPS. This proves that using two inexpensive, low-FPS cameras can achieve similar performance to an expensive, high-FPS camera, presenting a cost-effective alternative.

Key words : Dual cameras(듀얼카메라), FPS(초당 프레임), Autonomous driving(자율주행), Frame enhancement(프레임 강화), Up-sampling(샘플링 향상)

Nomenclature

FPS : frame per second
ROS : robot operating system
RMS : root mean sqaure

1. 서 론

자율주행 차량은 전방의 장애물이 있다고 판단하였을 시 회피하거나 신호등의 교통정보를 확인하여 정지 및 출발하는 등 차량 주변 환경을 인식하여 현재 상황에 맞는 주행 제어 동작을 위해 자율주행 기능이 구현되지 않은 차량에 비해 다양한 종류의 센서들을 적용하고 있다. 대표적으로 카메라, RADAR, LiDAR 및 초음파 센서들이 있으며, 이 센서들을 활용하여 Level 2 수준의 반자율주행 기술은 이미 대부분의 차량 제조사의 양산 차량에 적용이 완료된 실정이다. 현재 자율주행 차량은 종래의 인지 - 판단 - 제어 과정의 설계 방식을 벗어나 E2E(End to End) 설계 방식을 적용하는 등 기술 변화가 빠르게 일어나고 있으며 이러한 새로운 기술 적용을 위해 앞서 소개한 여러 센서들 중 카메라 센서의 역할은 더욱 커지고 있다. 카메라 센서의 중요성이 증가함에 따라 자율주행 차량을 위한 고성능 카메라 제품들이 출시되고 있다. 고성능 카메라는 고해상도 이미지, 뛰어난 감도, 자동 초점 기능과 함께 높은 FPS를 제공하여 도로와 장애물을 실시간으로 정확하게 인식할 수 있도록 도와준다. 특히, 자율주행 차량이 다양한 주행 환경에서 발생할 수 있는 돌발 상황을 명확하게 인지하려면 고성능 카메라의 도입이 필수적이다.[1)]

자율주행 기술의 안전성과 신뢰성을 높이는데 사용되

*Corresponding author, E-mail: hojin.jung@ut.ac.kr

는 카메라 센서는 하드웨어 성능 향상 뿐만 아니라 소프트웨어 성능 향상 또한 동시에 이루어지고 있는데, 최근 각광받는 인공지능 기술 적용을 통해 이미지 데이터를 분석하여 객체를 더욱 정교하게 인식할 수 있게 되었다.[2] 이와 같이 자율주행 기술의 안전성과 신뢰성을 높이기 위해서는 우선 고가의 카메라 센서를 통해 양질의 이미지 데이터를 획득한 이후에 인공지능 기법과 같은 고도화된 알고리즘으로 이미지 처리를 해야 차량 주변의 장애물이나 보행자, 도로 표지판 및 신호등과 같은 다양한 차량 주변 환경 정보들을 명확하게 인식할 수가 있다.

카메라 성능을 판별하는 기준은 크게 두가지로 구분할 수 있는데, 첫번째는 카메라 이미지의 해상도 이며 두번째는 카메라의 FPS이다. 해상도는 이미지 데이터에서 주변 상황에 대한 정보들을 분류하고 각 정보들에 대한 판단 정확성을 향상시키는데 직접적인 영향을 주는 요인이다. FPS는 이렇게 분류된 정보의 실시간성을 향상시킬 수 있는 요인이다. 낮은 FPS의 카메라를 사용하면 카메라에서 다음 이미지 데이터가 들어오기 이전까지는 과거의 데이터로 현재 상황을 판단하게 되므로, 카메라 기반의 자율주행 차량 상용화를 위해선 고해상도 이미지 데이터 뿐만 아니라 높은 FPS 데이터 확보 또한 필요하다. 특히, 저속 자율주행 차량이 아닌 고속 자율주행 차량 상용화를 위해서는 이미지 해상도 향상보다 FPS를 향상시키는 것이 더욱 중요하다.

그러나 성능이 높은 카메라일수록 가격 또한 상승하는 경향이 있다. 고해상도와 높은 FPS를 동시에 제공하는 카메라의 도입은 차량 구매비용을 증가시키게 된다. 특히, 자율주행 기술이 상용화되기 위해서는 가격 경쟁력이 중요한 요소로 작용하기 때문에, 이러한 고비용 카메라 센서의 도입은 큰 도전 과제가 된다. 따라서, 효율성과 경제성을 동시에 고려할 수 있는 대안이 필요하다.[3]

최근에는 이러한 한계를 극복하기 위해 단일 센서 사용에만 의존하지 않고 카메라 센서를 LiDAR, RADAR, IMU 등의 다른 센서와 융합해서 사용하는 기술이 주목받고 있다.[4] 각 센서는 서로 다른 특성과 장점을 지니는데, 카메라는 색상 및 형태 정보를, LiDAR는 정밀한 거리와 공간 구조 정보를, RADAR는 악천후나 야간 환경에서도 안정적인 속도와 거리 인식을 제공한다. 따라서 다중 센서 융합 방식을 사용하면 개별 센서의 한계를 보완하고, 고가의 단일 센서를 사용하는 것보다 적은 비용으로 주변 물체 인지 및 주행 상황 예측 및 판단의 정확도와 강건성을 향상시킬 수 있다.[5] 특히, 카메라-LiDAR 센서 간 융합은 이미지 기반 객체 분류 능력과 3D 위치 정확도를 동시에 확보할 수 있어 이미 자율주행 기술의 핵심 솔루션으로 활용되고 있다.[6,7]

이와 더불어, 센서 융합뿐 아니라 카메라 자체의 데이터 처리 및 프레임 생성 방식 개선 역시 객체 검출 성능 향상에 중요한 역할을 할 수 있다. 최근 연구에서는 이를 위한 방안으로 프레임 보간(Frame interpolation) 기법이 주목받고 있다.[8] 프레임 보간은 실제 촬영된 연속 프레임 사이에 새로운 중간 프레임을 가상으로 생성하여 간접적으로 FPS를 향상 효과를 통해 카메라 영상 데이터를 개선하는 기술이다. 이를 통해 고가의 높은 FPS 카메라를 사용하지 않고도 이와 유사한 효과를 얻을 수 있으며, 입력 영상의 연속성이 강화되므로 객체 인식 알고리즘의 검출률 향상에 기여할 수 있다.

그러나 프레임 보간 기법은 몇 가지 한계를 지닌다. 복잡한 움직임이나 고해상도 영상에서는 보간 과정에서 Ghosting, Blur, Halo와 같은 인공적인 왜곡이 발생할 수 있으며, 이는 객체의 경계가 흐려지거나 실제 존재하지 않는 픽셀이 생성되는 문제를 야기한다. 또한, 딥러닝 기반 보간 알고리즘은 대규모 연산 자원을 요구하여 연산 지연 및 전력 소모가 증가하며, 실시간 응용 환경에서는 즉각적인 처리가 어렵다는 제약이 따른다. 더 나아가, 학습 데이터에 대한 의존성이 높아 특정 장면 (저조도, 급격한 회전 동작)에서는 보간 품질이 저하될 수 있으며, 고품질 보간을 위한 복잡한 모델은 경량화 및 실시간성 측면에서 한계를 드러낸다.

한편, 저 FPS 카메라 여러 대를 시간적으로 분산 구동하여 고 FPS 영상을 구성하는 카메라 배열 기반 시간 해상도 향상 연구도 보고된 바 있다. Wilburn 등은 다수의 저가 CMOS 센서를 조밀 배열로 구성하고 각 센서의 촬영 시점을 위상차로 제어하여 고속 영상을 생성하는 시스템을 제시하였으며, 카메라 간 기하 · 방사 특성 차이를 보정하기 위해 정합 및 보정 절차의 필요성을 함께 논의하였다.[9] 또한 Shechtman 등은 복수의 저프레임률 영상을 결합하여 시공간 초해상도(Space-time super-resolution) 를 최적화 문제로 정식화함으로써 시간 해상도 향상을 시도하였다.[10] Agrawal 등은 다중 카메라의 노출 구간을 설계하는 Coded sampling을 통해 N대의 저속 카메라로 N배 수준의 시간 해상도 복원을 목표로 하였고, 복원 과정의 잡음 및 재구성 안정성 관점에서 샘플링 전략을 제안하였다.[11] 최근 Lu는 비동기 카메라 배열에서 시간 오프셋으로 취득된 프레임을 타임스탬프 기준으로 재배열하고, 기준 시점으로의 변환을 통해 고속 비디오를 생성하는 방법을 보고하였다.[12] 다만, 상기 연구들은 다수 카메라의 정밀 보정 및 영상 합성 품질을 주요 목표로 하며, 정합 오차 및 재구성 연산이 결과에 영향을 줄 수 있다. 이에 비해 본 연구는 자율주행 제어 적용 관점에서, 두

대의 저비용 웹캠을 목표 주기의 절반 수준으로 비동기 구동시켜 시간축에서 프레임을 교차 배치함으로써, 복잡한 보간·복원 과정 없이 실제 촬영 프레임만으로 FPS 향상 효과를 확보하고자 한다.

본 연구에서는 이러한 단점을 보완하기 위하여, 프레임 보간으로 생성한 가상의 중간 프레임을 사용하는 방식이 아닌 실제 듀얼 카메라 구성을 통해 물리적 시간차를 갖는 프레임을 직접 획득하는 방안을 제안한다. 즉, 두 대의 카메라를 서로 비동기화하여 동일 장면에 대해 시차가 있는 영상을 수집함으로써, 보간 과정에서 알고리즘 오류로 발생할 수 있는 영상 왜곡의 위험성을 극복하고 또한 영상 데이터 처리를 위한 연산 부담까지 감소시키면서 FPS를 향상시키는 효과를 얻고자 한다. 이는 프레임 보간법의 한계를 뛰어 넘어, 가상의 데이터가 아닌 실제 데이터시간 해상도를 확보할 수 있는 실용적이면서도 신뢰성 있는 해결책이 될 것이다.

듀얼 카메라는 물리적으로 서로 다른 위치에 장착되므로 동일 시각에 획득한 영상이라도 시점(Viewpoint)이 완전히 일치하지 않는다. 따라서 두 카메라 프레임을 교차 배치하여 시간 해상도를 높이는 경우, 단일 고속 카메라가 제공하는 동일 시점에서 공간적으로 일관된 연속 영상과 완전히 동일하다고 볼 수는 없다. 본 연구는 이러한 한계를 전제로 하며, Optical flow, 정밀 추적(Tracking) 등 공간 일관성에 민감한 알고리즘 성능을 직접적으로 대체하는 것을 목표로 하지 않는다. 대신, 실시간 제어 적용 관점에서 중요한 프레임 간격의 균일성, FPS 변동성, 과주기율(Over period rate) 등 시간 특성 지표를 중심으로 제안 기법의 효용성을 정량적으로 검증한다.

본 논문의 구성은 다음과 같다. 2장에서는 FPS 향상을 위한 기본 아이디어 및 관련 함수, 카메라 통신 방식 및 비 동기화 알고리즘에 대해 설명한다. 3장에서는 실험에 사용한 카메라 및 PC의 제원과 실험방법에 대해 설명한다. 4장에서는 실험을 통해 얻은 데이터로 제안한 방법에 대한 수치적 성능 결과 및 분석을 제시한다. 마지막으로 본 연구의 결론과 향후계획에 대해 설명한다.

2. 기술적 배경

2.1 카메라

웹캠은 USB 포트를 통해 이미지를 디지털 데이터로 변환하여 컴퓨터로 전송하는 장치이다. 이 과정에서 웹캠은 촬영 영상을 실시간으로 압축 및 인코딩하여 전송 효율을 높인다. 웹캠에서 전송된 디지털 이미지 데이터는 자율주행 시스템이 주변 환경을 인식하고 분석하는 데 활용된다.

Table 1 Camera price comparison

	Model	Logitech C920	Logitech StreamCam
Webcam	Pixel	2MP	2MP
	FPS	30	60
	Price	KRW 73,630	336,900

본 연구에서는 고 FPS 카메라 적용 시 발생하는 비용 부담을 고려하여, 시장 가격 기반 비교를 통해 웹캠 기반 접근의 경제성을 정량적으로 확인하고 이를 Table 1에 정리하였다. 웹캠 시장에서는 일반적으로 FPS가 증가할수록 가격이 상승하는 경향이 나타났다.

이에 따라 저 FPS 웹캠 2대 구성은 고 FPS 웹캠 1대 대비 비용 측면에서 유리한 대안이 될 수 있었다. 또한 웹캠은 양산형 제품으로 공급 범위가 넓고 접근성이 높아, 동일 조건의 장비를 비교적 쉽게 확보할 수 있다. 더불어 제조사 및 제품군에 따라 세부 가격 차이는 존재하므로 엄밀한 비교를 위해서는 동급 사양 제품군의 가격을 다수 수집하여 평균값을 산출하는 절차가 필요하다. 다만 웹캠은 대중화된 시장에서 가격이 비교적 안정화되어 있어 급격한 가격 변동이 제한적이며, 동일 FPS 구간 내에서는 가격 분포가 크게 이탈하지 않는 경향을 보이므로, 본 연구의 비용 비교 결과는 대표성 있는 수준에서 유사한 경향을 유지할 것으로 판단하였다. 실제로 동일 판매처 기준으로 비교했을 때, 동급 사양 제품 간 가격 차이는 30 FPS의 경우 2만원 내외 수준이며, 60 FPS의 경우 8만원 내외로 확인되었다. 따라서 본 연구에서는 비용 비교의 해석이 비교적 명확하고 접근성이 높은 웹캠 환경을 기준으로 실험을 수행하였다.

본 연구에서는 30 FPS 웹캠 두 대를 활용하여 이미지 데이터 획득을 위한 통신을 수행하였다. 각 카메라는 초당 30장의 이미지를 전송할 수 있으므로, 두 카메라를 병렬로 운용할 경우 초당 총 60장의 이미지 데이터 확보가 가능하였다. 또한 두 카메라의 전송 동작 시점을 서로 다르게 설정할 수 있다면, 두 카메라의 프레임이 시간축에서 교차(Interleave)되어 단일 60 FPS 카메라와 유사한 시간 해상도의 연속 데이터 흐름을 형성할 수 있을 것으로 예측하였다. 즉, 두 개의 30 FPS 카메라가 서로 다른 시점에 촬영한 영상을 전송함으로써, 고가의 고프레임 카메라 적용 없이도 프레임 연속성 향상을 기대할 수 있었다.

2.2 스레드 함수(Thread function)

본 연구에서는 자율주행 시스템의 효율적인 이미지 데이터 송수신을 위해 독립적인 스레드 함수를 활용하였다.[13)]

스레드 함수를 사용함으로써, 각 웹캠에서 전송되는 이미지 데이터를 별도의 스레드에서 병렬로 처리할 수 있게 되었으며, 이는 데이터 수집 및 처리 과정에서 발생할 수 있는 지연을 최소화하고, 실시간으로 환경 인식을 수행할 수 있도록 하였다. 특히, 이러한 구조는 두 개의 카메라로부터 수집된 프레임을 시간적으로 분리하여 처리할 수 있는 기반을 제공함으로써, 서로 다른 타이밍에 수신된 비동기화 이미지 데이터를 생성하고 활용하는 데에도 중요한 역할을 하였다.

2.3 ROS

ROS는 현대 로봇 개발의 핵심 플랫폼으로 자리 잡고 있으며, 다양한 로봇 시스템과 인공지능 응용 프로그램에서 필수적인 통신 메커니즘을 제공한다. ROS의 통신 구조는 로봇 소프트웨어의 모듈화 및 재사용성을 촉진하며, 노드 간의 효율적인 데이터 전송을 가능하게 한다. 특히, 이러한 통신 구조는 로봇 및 AI 응용 프로그램의 개발에 있어 필수적인 요소로 작용하며, 여러 모듈 간의 원활한 상호작용을 통해 시스템의 성능과 효율성을 극대화하는 데 기여한다.

본 연구에서도 이러한 ROS의 구조적 장점을 활용하여, 이미지 처리, 객체 인식, 센서 데이터 통신 등 각 기능을 독립적인 노드로 구성하고 상호 연동외에 데이터 전송에 사용함으로써 전체 시스템의 안정성과 확장성을 향상시켰다.[14)]

2.4 비 동기화 알고리즘

본 연구에서는 두 대의 카메라가 독립적으로 동작하면서 실시간으로 프레임을 수집하도록 시스템을 설계하였다. 이를 위해 Python의 멀티 스레딩을 적용하여, 1번 카메라와 2번 카메라가 각각 독립된 스레드에서 프레임이 획득되도록 구성하였다. 이와 같은 구조는 두 카메라가 병렬적으로 동작할 수 있도록 하여, 프레임 수집 지연을 최소화하고 전체 시스템의 실시간성을 향상시키는 데 기여하였다.

PC 환경에서 ROS 기반으로 멀티스레드 구조를 구성하고 두 개의 카메라 스레드(Camera_1, Camera_2)를 병렬로 실행하도록 설계하였다. Camera_1 스레드는 실행과 동시에 첫 프레임(Frame_1)을 획득한 뒤, ROS를 통해 프레임 데이터를 전송하도록 구현하였다. 이후 각 루프에서 처리 완료 시점부터 목표 주기까지 남은 시간만큼 대기하는 Term 구간을 수행한 뒤 다음 프레임(Frame_3, Frame_5…)을 획득한다.

여기서 Term은 카메라 스레드의 루프 주기가 일정하도록 유지하기 위한 주기 제어 구간으로, 프레임 처리 시간에 따라 가변적으로 대기 시간을 조절하는 단계이다.

Camera_2 스레드는 Camera_1과 동일한 주기 제어 구조를 사용하되, 두 카메라 스트림이 시간축에서 교차되도록 시작 시점에 초기 대기를 부여하였다. 즉, Camera_2는 프로그램 시작 직후 즉시 프레임을 획득하지 않고 Wait(0.016 s)만큼 대기한 뒤 첫 프레임(Frame_2)을 획득하며, 이후에도 Term 구간을 통해 일정 주기를 유지하면서 프레임(Frame_4 …)를 순차적으로 획득한다. 최종적으로 두 스레드에서 획득·전송된 프레임 시퀀스는 통합하여 이미지 데이터 및 로그에 시간 순서대로 기록함으로써, 단일 카메라 대비 촘촘한 시간 샘플링을 갖는 영상 스트림을 구성하였다. 이를 요약하자면 Fig. 1과 같다.

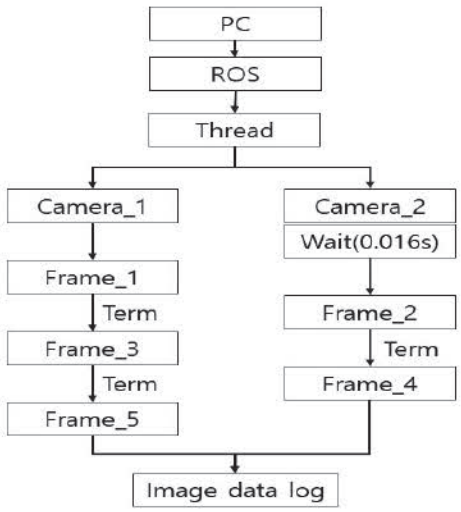


Fig. 1 Asynchronous algorithm

3. 시스템 구성 및 실험 방법

3.1 시스템 구성

3.1.1 실험 PC 구성 및 제원

본 연구에서 제안한 방식을 검증하고자 Desktop PC와 Mini PC에 저화질과 고화질의 듀얼 카메라를 통하여 실험을 진행하였다. 카메라와 PC의 제원은 각각 Table 2,3에 나타내었다.

Table 2 Desktop PC and mini PC specifications

	Desktop PC	Mini PC
CPU	Intel i7-13700F	6-core arm cortex-A78AE v.8.2
GPU	RTX 4060 Ti	Ampere architecture with 1024 CUDA cores and 32 tensor cores
Memory	48 GB	8 GB

Table 3 Low-definition camera and high-definition camera specifications

	Low-definition camera	High-definition camera
Resolution	640 × 480	1920 × 1080
Camera angle of view(Degree)	72.4	80
FPS	30	30
Manufacturer	Abko	Abko

3.1.2 통신 주기 설정

본 연구에서는 60 FPS의 카메라 성능을 갖는 통신을 목표로 하였기에, 통신 주기는 다음과 같이 설정하였다.

$$1/60 \approx 0.01667[s] \tag{1}$$

$$0.0166667[s]*1000[ms] \approx 16.67[ms] \tag{2}$$

따라서, 첫 번째 카메라에서 이미지 데이터를 전송한 뒤 16.67 ms 후에 두 번째 카메라에서 데이터를 전송하도록 설정하고 이 과정을 반복하여 60 FPS의 카메라처럼 동작할 수 있도록 하였다.

3.2 실험 방법

3.2.1 측정 방법

본 논문에서 제안한 비동기화 프레임 획득 방식의 객관적 검증을 위해, 30 FPS 카메라 2대에서 1초 동안 수집되는 약 60장의 이미지 데이터가 동일 시점의 중복 프레임이 아니라, 시간축에서 서로 다른 시점에 획득된 프레임임을 확인할 필요가 있다. 이를 위해 Desktop PC에서 ROS 기반으로 두 개의 스레드를 병렬로 실행하여 동일한 카메라 시스템에서 두 영상 스트림을 독립적으로 수집하였으며, 각 스레드는 각기 다른 이미지 데이터로 저장하도록 구성하였다.

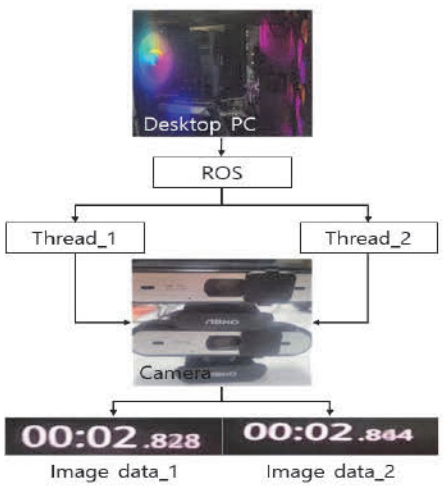


Fig. 2 Schematic diagram of data logging for dual camera

측정은 타이머가 표시되는 화면을 촬영하는 방식으로 수행하였다. 비동기화 알고리즘이 적용된 동작 환경에서 약 30초 동안 이미지를 저장하면서, 각 프레임 내부에 기록된 로그를 비교하여 두 스트림에서 저장된 이미지가 시간적으로 서로 겹치지 않고 교차되어 수집되는지를 확인하였다. 이를 요약하면 Fig. 2와 같다.

3.2.2 고려사항

일반적으로 카메라는 초당 프레임을 일정한 묶음 단위로 전달하는 방식을 사용하기 때문에, 이러한 설정만으로는 본 연구에서 제안하는 프레임 강화 효과를 구현하기 어렵다. 따라서 두 카메라의 영상 정보를 교차 수신할 수 있도록, 스레드 기반 통신 구조를 적용하여 각 카메라의 이미지 데이터를 병렬 처리함과 동시에 비동기화를 수행하였다. 이를 통해 약 17 ms의 시간 차이를 두어, 두 카메라가 생성하는 프레임이 서로 중첩되지 않고 중간 프레임과 유사한 형식으로 배열될 수 있도록 하였다. 또한, 30 FPS 카메라의 경우 기본적으로 프레임을 묶음 단위로 수신하는 동작을 보이므로, 이를 프레임 단위 수신 방식으로 설정을 변경하여 30프레임 전체가 아닌 개별 프레임 단위로 직접 처리할 수 있도록 하였다.

4. 실험 결과

4.1 Desktop PC

30초 동안 진행된 실험 결과는 Fig. 3와 Fig. 4에 각각 나타내었다.

먼저 Fig. 3은 Desktop PC 환경에서 고화질 카메라와 저화질 카메라의 시간차 변화를 나타낸 그래프이다. 그래프를 살펴보면, 고화질 카메라는 평균적으로 60 FPS 기준에 해당하는 약 16.67 ms 부근을 중심으로 동작하고 있으나, 순간적으로 통신 주기가 길어지거나 짧아지는 불규칙한 변동을 보였다. 특히 구간별로 진폭이 커지거나 줄어드는 패턴이 반복적으로 관찰되었으며, 이는 고화질 카메라의 경우 영상 처리 및 전송 과정에서 간헐적인 지연이나 버퍼링 현상이 발생할 가능성을 시사한다. 반면, 저화질 카메라는 고화질 카메라와 달리 시간 차의 편차가 비교적 작게 분포하였고, 전반적으로 일정한 간격을 유지하면서 보다 안정적으로 통신이 이어지는 모습을 확인할 수 있었다. 이러한 결과는 해상도의 차이가 실시간 전송 안정성에 직접적인 영향을 미칠 수 있음을 보여준다.

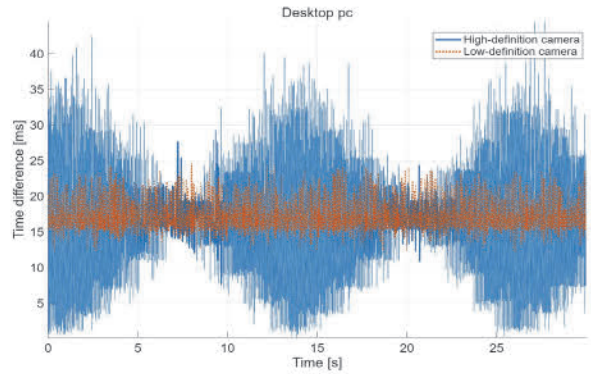


Fig. 3 Time difference comparison of high definition and low definition camera in desktop PC

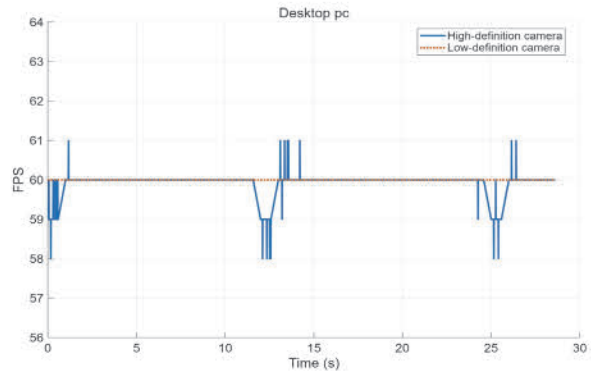


Fig. 4 FPS comparison of high definition and low definition camera in desktop PC

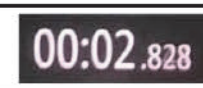
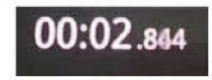
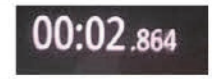
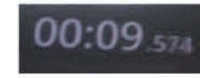
다음으로 Fig. 4는 동일한 실험 구간에서 고화질 카메라와 저화질 카메라의 FPS를 비교한 그래프이다. 분석 결과, 두 카메라 모두 목표치였던 60 FPS를 기준으로 동작하는 것을 확인할 수 있었다. 그러나 세부적으로 살펴보면, 저화질 카메라의 경우 모든 구간에서 60 FPS 유지하는 반면, 고화질 카메라는 평균적으로 목표 FPS에 도달하였음에도 불구하고 특정 시점 및 1초 내의 짧은 구간에서 ±2 FPS 수준의 변동이 발생하였다. 이는 고화질 카메라가 높은 해상도의 데이터를 처리하는 과정에서 순간적인 과부하나 불안정적 비동기화로 인해 FPS가 미세하게 변동하는 것으로 해석될 수 있다.

추가적으로, Table 3은 Desktop PC 환경에서 해상도별 통신 주기의 최대값, 최소값, 평균값 그리고 표준값을 정리한 결과를 제시한다. 또한, Table 4는 프레임 연속성 비교 결과를 나타낸 것으로, 저화질 카메라는 전 구간에서 일정한 간격을 유지하여 매끄러운 영상 흐름을 제공하였다. 반면, 고화질 카메라는 불규칙한 간격으로 인해 순간적인 끊김 현상이 발생하였다. 이러한 결과는 앞서 제

Table 3 Max, min, mean and RMS of communication time difference data in desktop PC

	Low-definition camera	High-definition camera
Max (ms)	24.53	44.49
Min (ms)	12.28	0.21
Mean (ms)	17.15	17.13
RMS (ms)	17.31	20.07

Table 4 Frame continuity comparison in desktop PC

Camera No.	Low-definition camera	High-definition camera
1		
2		
1		
2		
1		
2		

시한 통신주기 및 FPS 분석과 일관성을 가지며, 저화질 카메라가 실제 환경에서도 상대적으로 더 안정적인 출력 특성을 보인다는 점을 뒷받침한다.

종합하면, Desktop PC 환경에서 고화질 카메라는 목표했던 FPS 성능을 충족하지만 통신주기의 큰 변동성과 이로 인한 FPS의 순간 및 짧은 구간 편차가 나타나는 반면, 저화질 카메라는 안정적인 프레임 간격과 연속성을 보장하는 것으로 나타났다. 이러한 차이는 해상도 선택이 실시간 처리 안정성 및 프레임 일관성 측면에서 중요한 변수로 작용함을 의미하며, 실제 응용 환경에서는 성능 요구사항에 따라 적절한 해상도 조절이 필요함을 시사한다.

4.2 Mini PC

30초 동안 진행된 Mini PC 환경의 실험 결과는 Fig. 5와 Fig. 6에 각각 제시하였다.

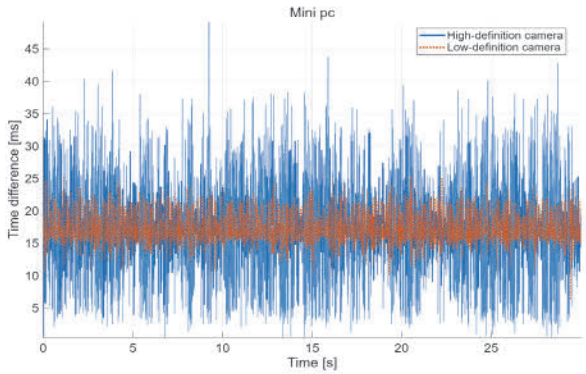


Fig. 5 Time difference comparison of high definition and low definition camera in mini PC

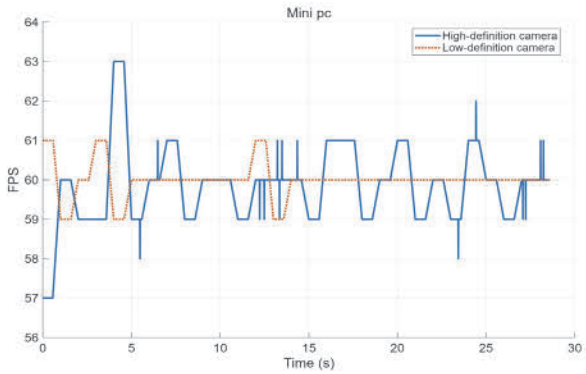


Fig. 6 FPS comparison of high definition and low definition camera in mini PC

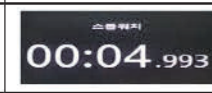
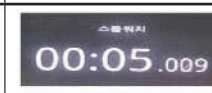
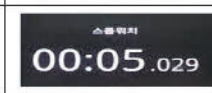
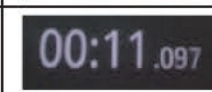
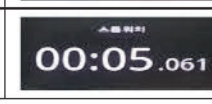
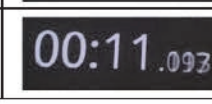
먼저 Fig. 5는 고화질 카메라와 저화질 카메라의 시간차를 나타낸 그래프이다. 고화질 카메라는 60 FPS 기준 주기인 16.67 ms를 중심으로 분포하였으나, 전반적으로 변동 폭이 매우 크며 간헐적으로 40 ms 이상까지 지연이 치솟는 현상이 나타났다. 이는 Mini PC의 제한된 연산 자원으로 인해 고해상도 영상 전송 과정에서 부하가 발생했음을 보여준다. 반면, 저화질 카메라는 Desktop PC 환경과 같이 안정적인 결과를 보였다.

다음으로 Fig. 6은 동일 구간에서의 FPS 변화를 나타낸 결과이다. 고화질 카메라는 평균적으로 60 FPS를 유지하였으나, 수초 내의 구간에 대한 ±3 FPS 수준의변동이 반복적으로 발생하는 현상이 나타났다. 이는 Mini PC 환경에서도 목표 FPS에는 도달하였지만 Desktop PC 환경 대비 안정성이 낮은 특성을 시사한다. 반면, 저화질 카메라는 고화질 카메라보다 FPS 변동 구간이 많지 않아 비교적 안정적인 동작을 보였다. 다만, Desktop PC 환경과 비교하면 FPS 변동 구간이 증가하였으므로 Mini PC의 성능 한계를 완전히 회피하지는 못한 것으로 보인다.

Table 5 Max, min, mean and RMS of communication time difference in mini PC

	Low-definition camera	High-definition camera
Max (ms)	25.32	49.21
Min (ms)	6.35	0.47
Mean (ms)	17.15	17.19
RMS (ms)	17.33	19.76

Table 6 Frame continuity comparison in mini PC

Camera No.	Low-definition camera	High-definition camera
1		
2		
1		
2		
1		
2		

추가적으로, Table 5는 Mini PC 환경에서의 해상도별 통신주기의 최대값, 최소값, 평균값 그리고 표준 편차 값을 정리한 결과를 제시한다. 또한, 연속성 실험 결과는 Table 6에 제시하였다. 저화질 카메라는 일부 구간에서 미세한 간격 차이가 발생하였으나, 전반적으로 일정한 프레임 간격을 유지하여 비교적 안정적인 연속성을 확보하였다. 반면, 고화질 카메라는 FPS 자체는 일정하게 유지되었지만, 프레임 간격이 불규칙하게 나타났다. 이는 해상도가 높을수록 Mini PC의 처리 부담이 커져, 결과적으로 연속성 측면에서 불안정성이 증가함을 보여준다.

종합하면, Mini PC 환경에서는 두 카메라 모두에서 전송 주기 변동성과 FPS 불안정성이 뚜렷하게 관찰되었다. 특히 저화질 카메라와 고화질 카메라 모두 목표 FPS를 달성하였으나 불규칙한 변동성이 커 신뢰성이 낮았으며, 이러한 균일한 비동기화의 성능 저하 현상은 저화질

카메라일 때보다 고화질 카메라일 때 더 두드러졌다. 이는 Mini PC 환경에서는 단순히 해상도를 낮추는 것만으로 안정성이 확보되지 않으며, 하드웨어 자원의 제약과 최적화 여부가 실시간 영상 처리 성능에 결정적인 영향을 미친다는 점을 시사한다.

4.3 저화질 카메라

Fig. 7은 저화질 카메라의 시간차를 Mini PC와 Desktop PC에서 비교한 결과이다. 두 PC 모두 전송 주기가 평균적으로 16.67 ms에 안정적으로 집중되며, 전 구간에서 일정한 주기를 유지하였지만, Mini PC는 평균 주기는 유사하였으나 변동 폭이 상대적으로 더 크고, 순간적으로 12 ms 이하로 내려가거나 24 ms 이상으로 치솟는 불규칙성이 관찰되었다. 이는 동일한 저화질 카메라라 하더라도 Mini PC에서는 연산 자원의 제약으로 인해 주기의 안정성이 다소 저하됨을 의미한다.

Fig. 8은 동일 구간에서의 카메라의 매 순간 FPS 변화를 비교한 결과이다. Desktop PC는 전 구간에서 60 FPS를 일정하게 유지하며 매우 안정적인 성능을 보였다. 이에 비해 Mini PC는 전반적으로 60 FPS를 유지하였으나, 특정 구간에서 ±1 FPS의 변동이 나타났다. 이는 저화질 카메라임에도 불구하고 Mini PC 환경에서는 순간적인 연산 부하로 인해 완벽한 FPS 안정성을 보장하기는 어렵다는 점을 확인 할 수 있었다.

4.4 고화질 카메라

Fig. 9는 고화질 카메라의 시간 차를 비교한 그래프이다. Desktop PC에서는 주기가 전반적으로 일정하지만, 순간적으로 진폭이 커지며 불규칙한 간격을 보이는 구간이 존재했다. 반면, Mini PC에서는 이러한 불규칙성이 더욱 크게 나타나며, 통신 지연이 40 ms를 초과하는 경우도 관찰되었다. 즉, 고해상도 데이터 전송의 경우 Mini PC가 안정성을 유지하기에는 한계가 뚜렷하게 드러나는 것으로 볼 수 있다.

Fig. 10은 고화질 카메라의 매순간 FPS 결과를 나타낸 그래프이다. 두 환경 모두에서 이미지 데이터의 소실은

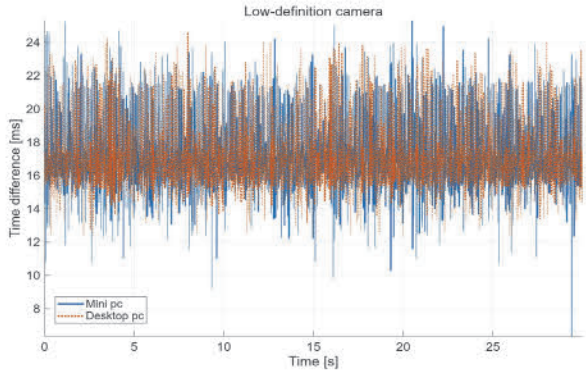


Fig. 7 Time difference comparison of low-definition camera between mini PC and desktop PC

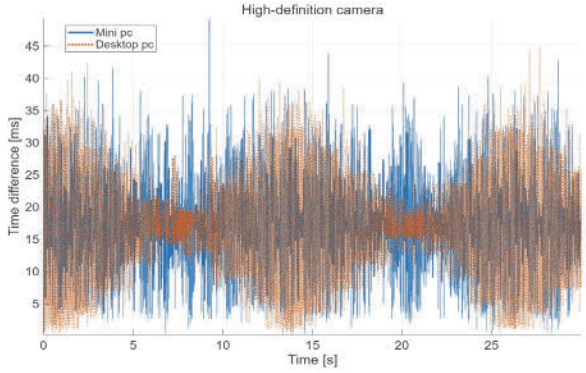


Fig. 9 Time difference comparison of high-definition camera between mini PC and desktop PC

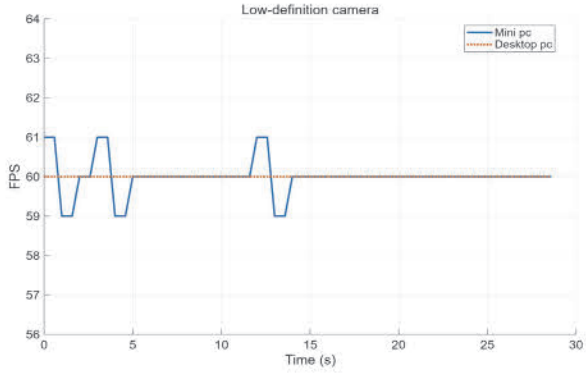


Fig. 8 FPS comparison of low-definition camera between mini PC and desktop PC

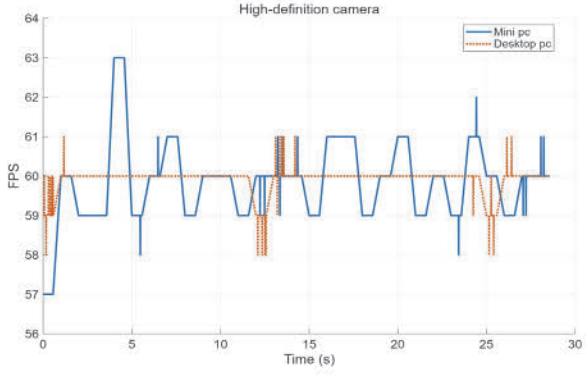


Fig. 10 FPS comparison of high-definition camera between mini PC and desktop PC

없었으며, Desktop PC는 60 ± 2 FPS, mini PC는 60 ± 3 FPS의 결과를 보여주었다. 세부적으로, Desktop PC가 순간적 변동을 제외하고 대부분의 구간에서 60 FPS의 값을 유지한 반면, Mini PC에서는 FPS의 감소와 증가 현상이 두드러지게 나타났다. 이는 Mini PC에서 고해상도 데이터를 처리할 경우 연산 지연과 동기화 불안정성이 더 크게 작용한다는 것을 의미한다. 보이는 바와 같이 고화질 카메라의 전송주기는 저화질 카메라의 경우보다 더 불규칙한 분포를 보였으며, 이는 실시간 FPS의 균일성을 확보하기 어렵다는 의미가 된다. 이에 따라 프레임 향상의 실질적인 효과를 설명하기 위해 단일카메라의 주기인 33.33 ms를 임계값(Threshold, μ)으로 설정하여 과주기(Over period) 현상을 분석하였다.

Fig. 11은 Mini PC와 Desktop PC 환경에서의 결과를 나타낸 것으로, 전체 구간에 대한 과주기율(Over period rate)은 Mini PC와 Desktop PC에서 각각 3.79 %와 3.95 %로 확인되었다. 즉 전체 데이터의 4 % 이내의 데이터를 제외하면 프레임 향상 효과를 기대할 수 있으므로 제안한 알고리즘의 효용성을 입증하였다. Fig. 12는 Fig. 11의

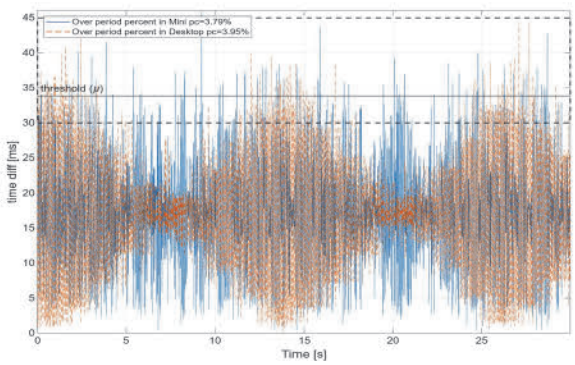


Fig. 11 Over period rate comparison of high-definition camera between mini PC and desktop PC

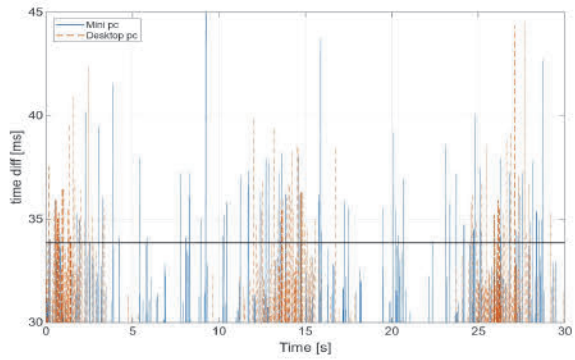


Fig. 12 Enlarged figure of figure 11

확대(Zoom-in) 결과로, 과주기율 분포의 세부 범위를 보다 명확히 나타낸다. 또한 Mini PC와 Desktop PC의 과주기율 차이는 크지 않았고 오히려 Mini PC에서 더 작게 나타나는 현상이 관측되어 제안한 방법의 실시간 차량 제어 적용에 대한 타당성까지 추가로 검증하였다.

5. 결 론

5.1 결론 및 요약

본 연구에서는 Desktop PC와 Mini PC 환경에서 저화질 및 고화질 카메라를 활용하여 30초간의 실험을 수행하고, 프레임 전송 주기, FPS, 그리고 프레임 연속성을 종합적으로 비교・분석하였다. 이를 Table 7에 정리하였다.

Desktop PC 환경에서는 저화질 카메라가 평균 17.15 ms의 안정적인 통신 주기와 높은 연속성을 유지하며 목표한 60 FPS에 근접한 성능을 구현하였다. 반면 고화질 카메라는 평균 FPS는 유사하게 유지되었으나, 통신주기의 편차가 크게 발생하여 프레임 간격이 불규칙하게 분포하였고, 이로 인해 영상의 연속성이 저하되는 현상이 확인되었다. Mini PC 환경에서 역시 비슷한 결과가 관찰되었는데, 저화질 카메라는 일정한 프레임 간격과 FPS 유지하여 상대적으로 높은 연속성을 보였으나, 미세한 변동이 발생하였다. 고화질 카메라는 프레임 간격의 불규칙성으로 인해 연속성이 저하되는 한계가 나타났다. 종합하면, Desktop PC와 Mini PC 환경에서 저화질 카메라가 연속성과 안정성 측면에서 우수했으나, Mini PC 환경에서는 약간의 변동이 발생하였다. 그리고 고화질 카메라에서는 Desktop PC 보다 Mini PC가 불규칙한 주기를 보였다. 이는 카메라 해상도와 하드웨어 자원의 상호작용이 실시간 영상 처리 성능에 중요한 영향을 미친다는 점을 보여주며, 실제 응용 환경에서는 사용 목적과 시스템 자원의 특성을 종합적으로 고려하여 해상도와 연산 자원을 균형 있게 선택해야 함을 시사한다.

제안한 방법의 실시간 제어기 탑재 가능성을 고려한

Table 7 Total comparison

	Low-definition camera		High-definition camera	
	Desktop PC	Mini PC	Desktop PC	Mini PC
Max (ms)	24.53	25.32	44.49	49.21
Min (ms)	12.28	6.35	0.21	0.47
Mean (ms)	17.15	17.15	17.13	17.19
RMS (ms)	17.31	17.33	20.07	19.76

Mini PC의 실험 결과에서는 매순간 균일한 FPS 확보가 어렵다는 점이 확인되었고, 특히 FPS의 높은 변동성으로 인해 일부 이미지 데이터는 단일카메라 사용할 때 기대할 수 있는 30 FPS보다 성능이 더 떨어질 수 있다는 결과도 확인하였다. 이러한 현상을 분석하기 위해 30 FPS 카메라의 시간차를 기준 임계치로 정하고 과주기 현상이 나타난 데이터의 비율을 수치화하였다. 분석 결과 Mini PC와 Desktop PC에서 과주기율은 모두 4 % 이내로 매우 낮게 나타나 제안한 방법의 우수성을 검증하였다.

본 논문에서의 주요 연구 성과를 요약하면 다음과 같다.

1) 저화질 카메라는 Desktop과 Mini PC 모두에서 평균 약 17 ms의 안정적인 통신 주기와 높은 연속성을 유지하며 목표 FPS에 근접한 성능을 보였다.
2) 저화질 및 고화질 카메라 모두 이미지 데이터의 손실 없이 실시간 FPS는 목표 FPS에서 ±3 FPS 이내의 값을 가졌다. 하지만 고화질 카메라에서 통신주기의 편차가 크게 발생하여 프레임 간격이 불규칙해지고 연속성이 저하되었다.
3) Mini PC 환경에서는 연산 자원 한계로 인해 저화질 카메라에서도 일부 변동성이 나타났으며, 고화질 카메라는 주기 지연과 불안정성이 더욱 심화되었다.
4) 고화질 카메라에서 일부 이미지 데이터는 단일카메라 사용때보다 통신주기가 더 저하되는 현상이 관측되었으며, 전체 이미지 데이터 중 4 % 이내의 데이터만 통신주기 저하 효과가 나타났다.

5.2 기대 효과 및 향후 계획

본 논문에서는 낮은 FPS를 갖는 두 개의 카메라를 사용하여 FPS를 높이는 방법에 대한 연구를 수행하였다. 자율주행 기술의 파급력과 중요성을 고려하였을 때, 본 연구의 성과가 자율주행 기술 상용화 시점을 앞당길 수 있는데 많은 기여를 할 것이다. 특히, 단순히 카메라의 FPS 향상만 시키는 것이 아닌 해상도 변화로 인한 성능 변동을 보정하고 안정적인 프레임 처리율까지 확보하도록 본 연구를 확장한다면 자율주행 시스템에서 카메라 기반으로 인지된 주변 환경 정보의 활용성이 더욱 향상될 것이다.

향후 연구에서는 실시간 제어기 적용성을 고려하여 Mini PC 실험결과에서 확인된 고해상도 이미지 전송 시 발생하는 불규칙한 통신주기와 과주기율로 표현되는 부분적인 통신주기 저하 문제를 개선할 수 있는 새로운 방법론을 적용할 계획인데, 프레임 동기화 및 부하 분산 기법을 적용하여 카메라 데이터가 해상도와 상관없이 일정한 속도와 안정성을 유지할 수 있게 한다면 상당한 성능 개선이 이뤄질 것으로 기대된다.

또한, 물리적으로 분리된 카메라 간 시점 차이로 인해 나타날 수 있는 공간적 비일관성에 대해서도, 향후 연구에서 그 영향과 특성을 추가로 검토하며, 필요 시 외부 파라미터 기반 정합, 평면 가정에 기반한 호모그래피/BEV 정합 등 기하 보정 기법의 적용 가능성도 함께 고려할 것이다. Optical flow, 객체 Tracking 등 공간 일관성에 민감한 알고리즘에 대해서는 적용 범위와 한계를 확인하는 방향으로 진행하면서 듀얼 카메라뿐 아니라 LiDAR, RADAR, IMU와 같은 이종 센서까지 포함한 센서 융합 환경에서의 실험을 통해 카메라 기반 인지 성능뿐만 아니라 전체 인지 시스템의 안정성을 강화하는 방향으로 연구를 확장할 예정이다.

후 기

2026년 국립한국교통대학교 지원을 받아 수행하였음.

References

1) R. Raman, P. K. Sa and B. Majhi, “Occlusion Prediction Algorithms for Multi-camera Network,” Proceedings of the Sixth International Conference on Distributed Smart Cameras (ICDSC), IEEE, pp.1-6, 2012.
2) D. Lee and H. Kim, “Object Detection and Tracking Through Cameras and LiDAR Fusion in Autonomous Vehicle,” KSAE Annual Conference Proceedings, pp.690-693, 2020.
3) H. Tan, F. Zhao, W. Zhang and Z. Liu, “An Evaluation of the Safety Effectiveness and Cost of Autonomous Vehicles Based on Multivariable Coupling,” Sensors, Vol.23, No.3, p.1321, 2023.
4) Y. Cui, R. Chen, W. Chu, L. Chen, D. Tian, Y. Li and D. Cao, “Deep Learning for Image and Point Cloud Fusion in Autonomous Driving: A Review,” IEEE Transactions on Intelligent Transportation Systems, Vol.23, No.2, pp.722-739, 2022.
5) S. H. Lee and W. Y. Choi, “Enhancing Object Estimation by Camera-LiDAR Sensor Fusion Using IMM-KF with Error Characteristics in Autonomous Robot Systems,” IEEE Access, Vol.12, pp.197247-197258, 2024.
6) Z. Wang, X. Huang and Z. Hu, “Attention-based LiDAR-camera Fusion for 3D Object Detection in Autonomous Driving,” World Electric Vehicle Journal, Vol.16, Paper No.306, 2025.
7) K. S. Choi, Y. H. Sa, S. J. Kim, D. S. Kang and J. U. Lee, “A Study on the Time Synchronization Method of Camera-LiDAR Sensor for Object

Detection in Autonomous Vehicles," Journal of the Korea Academia-Industrial Cooperation Society, Vol.24, No.10, pp.747-754, 2023.

8) J. Heo, K. Yoon, S. Kim and J. Jeong, "Research Trends in Deep Learning-based Video Frame Interpolation," Broadcasting and Media Magazine, Vol.27, No.2, pp.51-61, 2022.
9) B. Wilburn, N. Joshi, V. Vaish, M. Levoy and M. Horowitz, "High-Speed Videography Using a Dense Camera Array," Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Vol.2, pp.294-301, 2004.
10) E. Shechtman, Y. Caspi and M. Irani, "Space-Time Super-Resolution," IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol.27, No.4, pp.531-545, 2005.
11) A. Agrawal, M. Gupta, A. Veeraraghavan and S. G. Narasimhan, "Optimal Coded Sampling for Temporal Super-Resolution," Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp.599-606, 2010.
12) S. Lu, "High-Speed Video from Asynchronous Camera Array," Proceedings of the IEEE Winter Conference on Applications of Computer Vision (WACV), pp.2196-2205, 2019.
13) C. J. Nam and C. H. Lee, "Implementation for Real-time Control of Cores and Threads on Multi-core Linux-based Systems," Journal of the Korea Contents Association, pp.3-4, 2023.
14) J. H. Jeong, W. S. Choi and S. G. Choi, "ROS-based Real-time Image and Timestamp Collection System," Proceedings of the Korea Institute of Communication Sciences Conference, pp.422-423, 2024.