논문 22-47-07-10 The Journal of Korean Institute of Communications and Information Sciences '22-07 Vol.47 No.07
https://doi.org/10.7840/kics.2022.47.7.978

객체 인식을 이용한 차량 모니터링 시스템

장 진 호•, 박 은 영*, 황 정 수*, 유 영 환°

Vehicle Monitoring System Using Object Detection

Jin-ho Jang•, Eun-young Park*, Jeongsoo Hwang*, Younghwan Yoo°

요 약

본 논문은 YOLOv5의 실시간 객체 인식을 이용하여 교통 상황을 모니터링하고, 신뢰성 있는 교통 정보를 수집하는 시스템을 설계한다. 설계된 시스템은 카메라 한 대와 AI 보드 한 대만을 사용하므로 이동 설치에 용이하고, 측정 장비 설치 시 전문인력을 필요로 하지 않으므로 기존 교통 정보 수집 시스템의 경제적 단점을 해결한다. 객체 인식 시에 YOLOv5를 사용하므로 순차적으로 필터를 움직이면서 영상을 처리하는 CNN 기법보다 속도와 정확도에서 높은 성능을 발휘한다. 따라서 신뢰도 높은 정보 수집이 가능할 것이며 수집된 정보를 자동으로 서버에 저장한다. 이 시스템을 구현함으로써 장래 교통량 추정, 도로 계획 및 관리에 필요한 자료 수집에 유용하게 활용할 뿐만 아니라 효율적인 실시간 교통량 파악 및 시스템 비용 절감을 기대할 수 있다.

키워드 : 실시간 객체 인식, 교통 정보 수집, YOLOv5, DeepSort, Ai 보드
Key Words : real-time object detection, traffic data-collection, YOLOv5, AI board

ABSTRACT

This paper designs a system that monitors traffic conditions and collects reliable traffic information using real-time object recognition of YOLOv5. The designed system uses only one camera and one AI board, making it easy to install on the move, and does not require professionals to install measuring equipment, solving the economic drawbacks of the existing traffic information collection system. Since YOLov5 is used for object recognition, it exhibits higher performance in speed and accuracy than CNN techniques that process images while moving filters sequentially. Therefore, reliable information collection will be possible, and the collected information is automatically stored in the server. By implementing this system, it is not only useful for estimating future traffic volume, collecting data necessary for road planning and management, but also expected to efficiently identify real-time traffic volume and reduce system costs.

Ⅰ. 서 론

매일 TV나 라디오에서는 실시간 교통 정보를 제공하고 있다. 이는 교통 정보가 일기예보처럼 사람의 일상생활을 계획하고 실행하는 데 중요한 요소로 작용하고 있다는 것을 의미한다. 따라서 신뢰도 높은 교통 정보를 수집하고, 그 정보를 적절하게 활용하는 것이 중요하다. 교통 정보를 생성하는 일반적인 방식은 도

※ 이 과제는 부산대학교 기본연구지원사업(2년)에 의하여 연구되었음.
• First Author : Pusan National University, Department of Information Convergence Engineering, jjh101101@pusan.ac.kr, 학생회원
° Corresponding Author : Pusan National University, Department of Information Convergence Engineering, ymomo@pusan.ac.kr, 종신회원
* Pusan National University, Department of Information Convergence Engineering, ey5321@pusan.ac.kr; wjdtn5629@gmail.com
논문번호 : 202201-016-D-RN, Received January 28, 2022; Revised April 28, 2022; Accepted May 9, 2022

로에 차량 검지기를 설치하여 교통량을 수집하고 이를 다양한 알고리즘에 적용하여 구간 단위의 교통 정보를 생성하는 것이다. 따라서 교통 정보의 정확도는 차량 검지기에 수집되는 자료의 정확도에 따라 좌우된다.

차량 검지기는 국내에 지능형 교통 시스템(Intelligent Transport Systems, 이하 ITS)이 도입된 이후, 도로 위의 센서 기술을 기반으로 하여 개발되었다. 기존 ITS 검지 체계에는 루프 검지기, 적외선 검지기, 영상 검지기 등이 있다. 이 중 영상 검지기는 국내에서 많이 사용되고 있는 차량 검지기 중 하나이다. 영상 검지기는 카메라로 교통의 흐름을 영상화하고 해당 영상에 검지 영역을 설정하여 차량이 검지 영역을 통과할 때 교통량을 측정하는 방식으로 동작한다[1].

현재 사용하는 영상 검지기는 검지 영역에서의 화소 값의 변화에 의해서만 차량 통과 여부를 판단하기 때문에 미세한 환경변화에 민감하게 반응한다. 특히 여러 대의 차량이 겹쳐서 운행한다면 뒤 차량은 잘 인식하지 못하여 정확도가 떨어진다. 또한 영상 처리 시, 순차적으로 필터를 움직이면서 이미지를 처리하는 인공신경망의 한 종류인 CNN (Convolutional Neural Network) 기법을 사용한다. 필터를 순차적으로 움직이면서 시간이 소요되므로 속도 측면에서 성능이 좋지 않다[2]. 또한 설치 및 유지 시 전문 인력이 필요하고 가격도 고가이기 때문에 경제적인 측면에서도 단점을 가진다.

따라서 본 논문에서는 이러한 기존 시스템의 문제점들을 개선하여 객체 인식 시 YOLOv5를 사용함으로써 교통 정보의 정확도와 측정 속도를 높이고, 카메라 한 대와 Jetson 보드 한 대만을 이용하여 교통량을 측정함으로써 경제적 단점을 보완하며, 어느 장소에서든 설치하고 사용이 가능한 교통량 모니터링 시스템을 제안한다. 본 논문에서 제안하는 시스템은 부산광역시 구서역 인근 제한 속도가 50km/h인 일반 도로에서 약 1시간 동안의 실험을 거쳤다. 차량 계수 알고리즘은 10분 동안 통과한 323대의 차량 모두를 정확히 계수하였고, 차량 속도 측정 알고리즘은 오차 평균은 1.99km/h을 기록했고 평균 절대비 오차(Mean Absolute Percentage Error, MAPE)는 5.54%를 보였다. 해당 결과는 평균 절대비 오차가 7.6%인 기존 연구[3]와 비교하여 약 2.1% 만큼 낮았다. 따라서 본 논문은 기존 시스템에서 경제적 단점을 보완할 뿐만 아니라 성능 또한 개선된 시스템을 제안한다.

논문 구성은 다음과 같다. 2장에서는 본 실험에서 객체 인식 기술로 사용한 YOLOv5에 대해서 간략히 설명한다. 또한 실시간 물체 추적과 관련한 기법인 DeepSORT 알고리즘과 서버와 Jetson 보드 간의 통신에 사용한 Socket 통신을 설명한다. 3장에서는 본 논문이 제시한 차량 모니터링 시스템에 대해 자세히 설명한다. 4장에서는 제안 방법의 성능을 평가하는 실험을 진행하고 그 결과를 분석한다. 마지막으로 5장에서는 결론과 향후 연구 방향을 간략히 서술한다.

Ⅱ. 배경 지식

2.1 YOLOv5

YOLOv5는 YOLO(You Only Look Once)의 5번째 버전으로서 단일 CNN 모델 하나로 특징 추출, Bounding box 계산, 클래스 분류까지 모두 수행한다. YOLO 모델에서는 하나의 이미지가 입력되면 그리드 셀로 분할하고 Bounding box를 생성한다. Bounding box에 신뢰 점수 Confidence Score와 객체가 해당 Class일 확률을 예측하여 각 그리드 셀에 부여한다. 이때 Confidence Score는 수식1과 같이 계산되며 해당 셀에 객체가 존재하지 않으면 0이 된다. 객체가 존재한다면 IOU 값을 갖는데, IOU는 교차 겹침 결합(Intersection Over Union) 이라는 뜻으로 실제로 객체가 존재하는 box와 객체가 존재할 것이라 예측한 box가 겹치는 부분의 비율을 의미한다. 이와 같이 계산하여 각 그리드 셀에 부여된 값들을 곱하고, 그 결과 값이 설정한 threshold 값 이상이면 해당 객체로 판단한다[2]. YOLOv5의 모델에는 s, s6, m, m6, l, l6, x, x6 버전이 있는데 ‘s’ 버전으로 갈수록 속도는 빠르지만 정확도가 떨어지고, ‘l’ 버전으로 갈수록 속도는 느리지만 정확도가 높다는 특징이 있다[4].

$$Confidence\ Score = \Pr(Obj) * IOU \tag{1}$$

2.2 DeepSORT 알고리즘

DeepSORT 알고리즘은 움직이는 객체를 추적하는 Object Tracking 기법 중 하나로, 딥러닝(Deep learning)과 SORT 기법을 합친 것이다[5]. SORT 기법에서는 이전 프레임에서 탐지된 객체의 위치 정보를 Kalman filter에 적용하여 해당 객체의 다음 위치를 예측한다. 다음 프레임에서는 현재 탐지된 객체와 이전 프레임에서 예측한 객체의 IOU distance를 계산한다. 계산된 값을 Hungarian 알고리즘에 적용한 후 동일 객체인지 판단하고 객체 정보를 업데이트한다[6]. 이러한 SORT 기법에 Feature vector를 추가로 반영한 것이 DeepSORT 알고리즘이다. Feature vector는

입력 이미지에 대한 전체 특징(feature)을 묘사할 수 있는 벡터로, 객체 외관을 대표하는 벡터이다. Feature vector를 적용한 DeepSORT 알고리즘은 SORT 기법 적용 시 나타날 수 있는, 객체가 가려지는 문제를 개선할 수 있다[7].

2.3 Socket 통신

Socket 통신은 서버와 클라이언트가 특정 포트를 통해 실시간으로 전송되는 양방향 통신이다. 서버가 클라이언트에게, 클라이언트가 서버에게 요청을 보낼 수 있으며 항상 연결을 유지하는 연결지향형 통신이기 때문에 실시간 통신이 필요한 경우 주로 사용한다. Socket 통신 방법 중에는 TCP 통신과 UDP 통신이 있다. TCP는 클라이언트와 서버가 연결된 상태에서 데이터를 주고받는 연결 지향형 프로토콜이다. UDP는 데이터를 주고받을 때 발신하는 측이 일방적으로 데이터를 보내는 방식인 비연결형 프로토콜이다. 두 통신 방식을 표2와 같은 차이를 보인다. 연결 지향형 프로토콜인 TCP는 3-way Handshake 과정을 거쳐 클라이언트와 서버를 연결하고, 연결이 완료되면 통신 선로가 고정되어 모든 데이터가 선로를 통해 순차적으로 전달된다. 반면 UDP는 이러한 연결과정을 거치지 않기 때문에 TCP보다 빠른 전송을 할 수 있지만 데이터 전송의 신뢰성이 떨어진다. 따라서 본 논문에서는 일대일 통신에 적절하고 데이터 신뢰성 확보에 용이한 TCP 통신 방식을 사용한다[8].

표 1. TCP-UDP 통신 비교 표
Table 1. TCP-UDP Comparison Table

	TCP	UDP
Connection	Connection-Oriented	Connection-Less
Packet Exchange	Segment	Datagram
Transmission Order	Orderd	Nor Ordered
Communication	1:1	1:1, 1:N, N:N
Reliability	Reliable	Unreliable
Speed	Low	High

Ⅲ. 시스템 설계 및 구현

3.1 시스템 구조

그림 1은 본 연구에서 설계한 전체 시스템 구조이다. 쉽게 이동 가능한 임베디드 플랫폼이면서도 고성능 AI 기능을 제공하는 NVIDIA 사의 Jetson 보드를

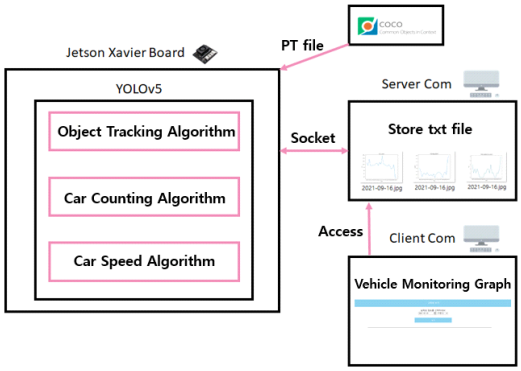

그림 1. 전체 시스템 구조
Fig. 1. Architecture of overall system

사용하는데, 그 중에서도 IoT 디바이스, 지능형 비디오 분석시스템 등의 환경에서 가장 강력한 딥러닝 성능을 발휘하는 Jetson Xavier NX Board 모델을 선택하였다. 측정 카메라로는 1080p 30fps의 해상도를 가진 웹캠을 사용하였다.

도로 위에 카메라와 보드를 설치하고 내장된 알고리즘을 실행하여 교통 데이터를 저장한다. 보드는 서버 컴퓨터와 통신하여 앞서 저장한 데이터를 서버 컴퓨터에 저장한다.

그리고 서버 컴퓨터에 저장된 차량 수 데이터, 차량 속도 데이터는 일별 단위로 그래프가 생성된다. 클라이언트는 웹으로 접속하여 전날까지의 교통량 그래프를 확인할 수 있다.

본 장에서는 객체 인식을 이용한 차량 모니터링 시스템에 구현된 알고리즘과 통신 방법을 설명한다.

3.2 객체 추적 알고리즘

YOLOv5만을 활용하여 객체 인식을 하면 연속된 프레임에 나타나는 같은 차량을 매번 독립적인 다른 차량으로 인식한다. 그러나 본 연구에서는 차량 속도를 측정해야 하기 때문에 움직이는 차량을 인식하여 시간, 거리 정보를 저장하여야 한다. 따라서 영상에서 해당 객체가 움직이는 path가 얼마나 그 전 프레임과 유사한지 알아낸 다음, 동일 객체라고 인식하게 되면 그 객체를 계속 추적하는 Object Tracking 기법을 활용하였다. Optical Flow[9]와 같은 OpenCV 라이브러리를 사용한 Object Tracking 기법은 AI 보드에 적용하기에 많은 메모리를 요구하여 메모리 부족 현상을 유발할 수 있다. 따라서 프레임 간 데이터 처리 시 간단한 Hungarian 알고리즘을 사용함으로 연산 처리량을 줄여 메모리 부족 현상을 해결하고, 프레임 간 동일 객체 판단 시 딥러닝 기반 알고리즘을 함께 사용하는

DeepSORT 알고리즘을 선택하여 정확도를 높였다.

3.3 차량 계수 알고리즘

YOLOv5 모델은 같은 차량임에도 크기의 차이가 크면 다른 객체로 인식한다. 따라서 일정한 크기 이상의 차량만을 인식하기 위해 차량 수 측정 알고리즘에서는 기준선을 설정한다. 그림 2와 같이 차량으로 인식된 Bounding box의 중점이 기준선 아래에 있으면 측정되는 방식으로 구현한다.

기준선 아래에서 같은 번호의 Bounding box가 두 번 이상 측정될 수 있다. 따라서 이전 프레임에서 측정된 번호는 배열에 저장하고 현 프레임에서 인식된 번호와 비교하여 중복으로 측정되지 않도록 한다. 예를 들어 그림 3과 같이 ID가 93인 차량이 여러 프레임에 걸쳐 기준선 아래에 있다고 인식되지만, 프레임이 바뀌어도 ID가 전 프레임에서 저장된 배열에 존재한다면 더 이상 ID를 저장하지 않는다. 측정된 값은 txt 파일에 시간과 1분 동안 지나간 차량의 수를 저장한다.

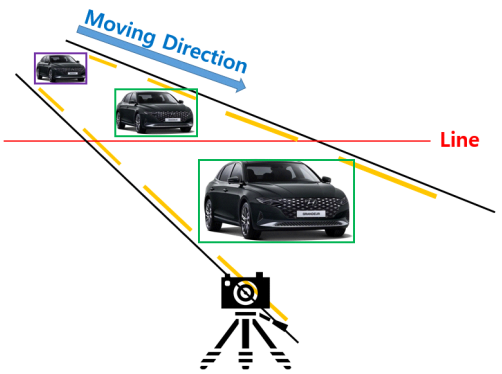

그림 2. 차량 수 측정
Fig. 2. Vehicle counting

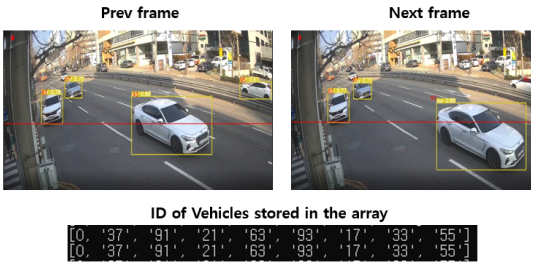

그림 3. 차량 ID 저장 방식
Fig. 3. Method to store vehicle IDs

3.4 차량 속도 측정 알고리즘

3.4.1 두 기준선 사이의 실제 거리 계산

차량 속도 측정 알고리즘은 수식 (2)에 각 값을 대입하여 계산하였다.

$$\text{차량 속도}(km/h) = \text{두 기준선 사이의 거리}(km)/\text{걸린 시간}(h) \quad (2)$$

이때 두 기준선 사이의 실제 거리를 알기 위해 그림 4와 같은 방식으로 계산하였다. 원리는 간단하다. 먼저 설치 후 카메라가 사람을 인식하면 Bounding Box의 중점 위치를 측정한다. 그 후 삼각대의 높이를 정해진 값(이 실험에서는 30cm)만큼 높였을 때 화면에 나타나는 Bounding Box의 위치가 달라진다. 카메라 위치를 높인 후 변동된 Bounding Box의 중점 위치를 측정하고 이전 위치와 비교하여 이동된 픽셀 수를 저장한다.

$$\text{두 기준선 사이의 실제 거리} : \text{두 기준선 사이의 픽셀 수} = 30cm : \text{이동된 중점의 픽셀 수} \quad (3)$$

이동된 픽셀 수와 두 기준선 사이의 픽셀 수를 수식 (3)에 대입하여 두 기준선 사이의 실제 거리를 알 수 있다.

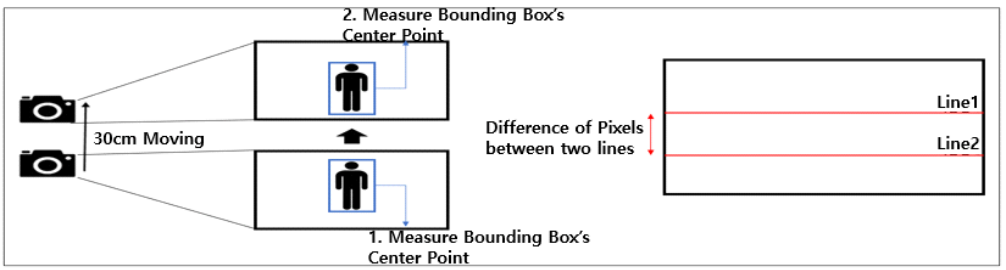

그림 4. 실제 거리 비율 계산 방법
Fig. 4. Calculating the ratio between the distance on monitor and the actual distance

3.4.2 차량 속도 측정 알고리즘

그림 5는 차량 속도 측정 알고리즘이 작동하는 흐름도이다. 먼저 그림 6과 같이 두 개의 기준선을 긋고 차량이 인식되기를 기다린다. YOLO 알고리즘으로 인해 차량이 인식되면 인식된 차량이 첫 번째 기준선을 지나갔는지 판단한다. 첫 번째 기준선을 Bounding box의 중점이 지나가면 DeepSORT 알고리즘으로 생성된 차량의 ID와 지나간 시간을 저장한다. 마찬가지로 두 번째 기준선을 지나치면 차량 ID와 지나간 순간의 시간을 저장한다. 그 후 수식2를 사용하여 두 기준선을 지나가는 데 소요된 시간과 기준선 사이의 거리로 속도를 계산한다.

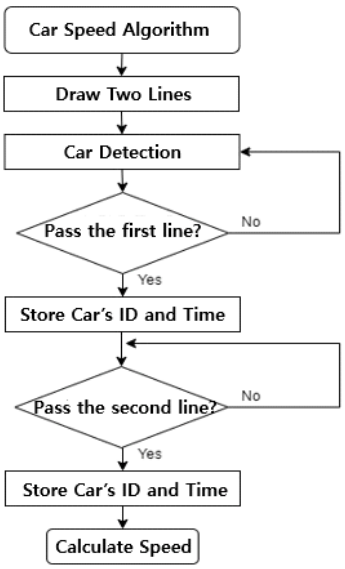

그림 5. 차량 속도 측정 알고리즘 흐름도
Fig. 5. Flow chart of car speed measuring algorithm

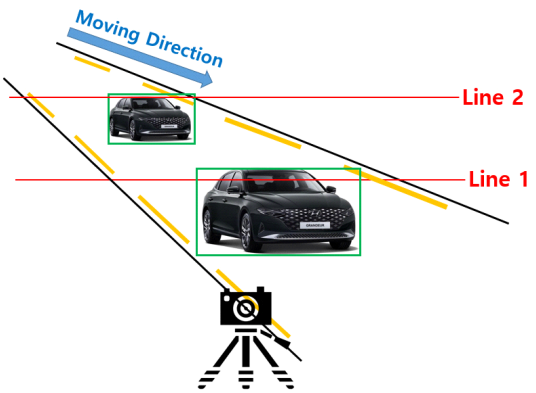

그림 6. 차량 속도 측정 방식
Fig. 6. Car speed measurement method

3.5 보드 - 서버 간 통신

본 논문에서는 실시간으로 Jetson 보드에서 처리되는 데이터를 지속적으로 전송해야 하며, 하나의 서버에 여러 클라이언트가 동시에 접속하지 않고 클라이언트인 Jetson 보드 하나와 서버 PC 하나 사이에 일대일 통신이 이루어지므로 Socket 통신 방식을 사용한다. 전송되는 데이터는 신뢰성이 높아야 하므로 Socket 통신 중에서 TCP 프로토콜 방식을 택한다.

그러나 TCP 프로토콜을 이용한 통신은 UDP 프로토콜 통신보다 전송속도가 느리다. 또한, 본 연구에서는 Jetson 보드에서 통신만 처리하지 않고 영상 처리 작업도 수행해야 하므로 3개의 멀티 스레드로 역할을 분할한다. 각 스레드의 역할은 아래 표 2와 같다.

통신을 통해 전송되는 데이터는 1분당 통행 차량 수, 각 차량의 속도와 과속 차량이 검지되었을 때의 영상 캡쳐 이미지이다. Jetson Board의 영상 처리 스레드인 첫 번째 스레드에서 처리된 영상의 이미지 데이터는 OpenCV 라이브러리를 통해 binary 형태로 암호화되고 임시로 queue에 삽입된다. 그 후 이미지 전송 스레드인 세 번째 스레드에서 큐에 삽입되어 있는 데이터를 꺼내어 서버로 전송한다. 서버에서는 전송받은 데이터를 복호화하여 이미지 파일로 저장한다.

표 2. 스레드 종류와 역할
Table 2. Types and Roles of Threads

Thread	Role
First Thread	Overall processing of captured videos
Second Thread	Send the measured number of vehicles and vehicle speed in the first thread to the server PC
Third Thread	Send vehicle image data when vehicle is overspeeding

Ⅳ. 실험 및 평가

그림 7는 실제 실험을 수행하는 모습이다. 실험은 부산광역시 금정구 구서역 인근에 위치한 육교에서 진행하였다. 실험에 사용된 보드는 Jetson Xavier NX 보드이며, 카메라는 1080p 30fps의 해상도를 가진 웹캠을 사용하였다. 또한 측정상황을 지속적으로 관찰하기 위해 모니터를 Jetson 보드에 연결하여 사용하였다.

4.1 차량 수 측정 알고리즘

그림 8은 차량 수 측정 알고리즘을 실행한 화면이

그림 7. 실험환경
Fig. 7. Experimental environment

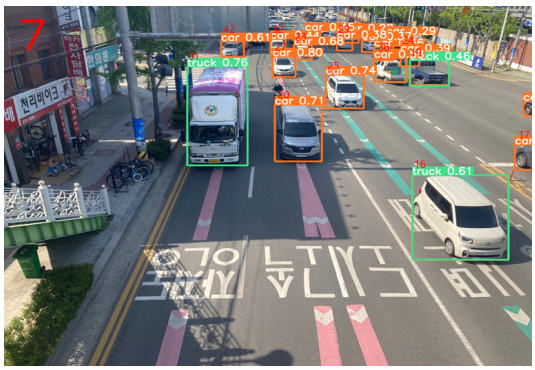

그림 8. 차량 수 측정 화면
Fig. 8. Vehicle counting monitor

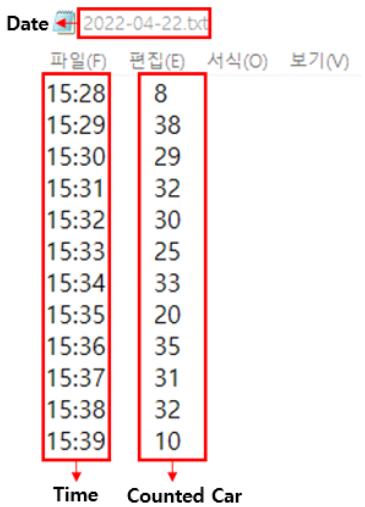

그림 9. 측정된 차량 수 저장 파일
Fig. 9. File of the number of vehicles

다. YOLOv5으로 인하여 차량이 'car'로 인식되고 주황색 box로 경계선이 그어졌다. box의 왼쪽 상단에는 DeepSort 알고리즘으로 인한 차량의 ID가 표시된다. 또한 그림 8의 왼쪽 상단에 보이듯 실시간으로 차량 수가 측정되어 지나간 차량의 수를 볼 수 있다.

그림 9는 차량 수 측정 알고리즘으로 지나간 차량의 수를 측정 후 저장되는 텍스트파일 화면이다. 측정된 날짜를 구별하기 위해 날짜별로 파일을 생성하고 1분 단위로 지나간 차량의 수를 저장한다.

4.2 차량 속도 측정 알고리즘

그림 10은 차량 속도 측정 알고리즘으로 지나간 차량의 속도를 측정 후 저장되는 텍스트파일 화면이다. 측정된 날짜를 구별하기 위해 날짜별로 파일을 생성하고 속도가 측정된 차량의 ID와 시간, 속도를 저장한다.

표 3은 차량의 실제 속도와 차량 속도 측정 알고리즘으로 측정된 속도를 비교한 표이다. 차량의 실제 속도는 레이더파의 도플러효과를 원리로 한 스피드건을 사용하여 측정하였다. 측정 된 차량의 오차 평균은 1.9897km/h을 기록했다. 수식 (4)를 사용하여 측정한 평균 절대비 오차의 평균은 5.53%을 기록했다. 기존 연구의 평균 절대비 오차값인 7.6%와 비교했을 때 약 2.1% 낮은 수치이므로 기존 연구보다 개선된 성능이다. 또한 그러나경찰청 관계자와 인터뷰한 기사글에 따르면 제한속도에서 10km/h를 초과하는 속도까지는 단속되지 않는다고 한다[10]. 따라서 모든 오차의 절댓

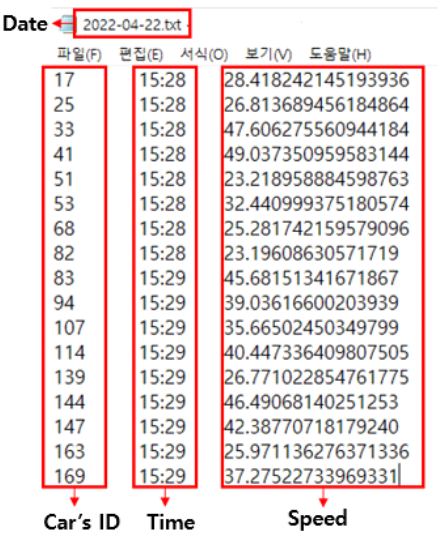

그림 10. 측정된 차량 속도 저장 파일
Fig. 10. File of each vehicle speed data

표 3. 속도 측정 결과표
Table 3. Speed measurement result table

Car ID	Real Speed(km/h)	Measured Speed(km/h)	|Error| (km/h)	MAPE (%)
17	28	28.4182	0.4182	1.49
25	27	26.8136	0.1864	0.69
33	44	47.6062	3.6062	8.19
41	48	49.0373	1.0373	2.16
…	…	…	…	…
183	40	40.6777	0.6777	1.69
207	48	50.0159	2.0159	4.19
213	44	43.8298	0.1702	0.38
235	52	50.6193	1.3807	2.65
293	45	46.4906	1.4906	3.31
307	53	56.6363	3.6363	6.86
345	61	62.2237	1.2237	2.01
Average			1.9897	5.53

값이 6km/h 미만인 본 실험 결과는 과속 단속에 사용되기 적절하다.

$$MAPE = \frac{100}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right| \quad (4)$$

$(n = \text{데이터 개수}, y = \text{실제값}, \hat{y} = \text{측정값})$

4.3 측정 데이터의 시각화

Socket 통신을 통하여 Jetson Board에서 받아온 차량 수 측정 텍스트파일과 속도 측정 텍스트파일을 서버 컴퓨터에 저장한다. 하루가 지나면 전날의 텍스트파일을 활용하여 그래프로 표현한다.

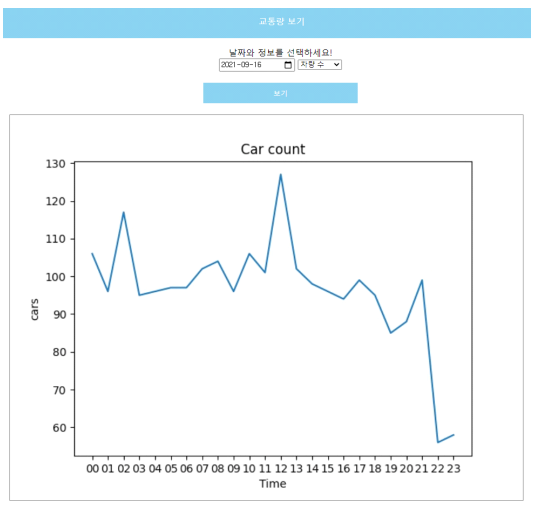

그림 11. 일별 차량 수 그래프 웹 화면
Fig. 11. Web screen capture of the number of vehicles of every day

그림 11은 클라이언트가 웹에 접속하여 확인할 수 있는 일별 차량 수 그래프 웹 화면이다. 확인하고 싶은 날짜를 선택하고 차량 수, 차량 속도 평균, 과속 차량 수 중 하나의 카테고리를 선택하면 해당하는 카테고리의 그래프가 나오게 된다. 예를 들어 2021년 9월 16일의 차량 수를 확인하면 그림 11과 같이 웹 화면에 나타나게 된다.

V. 결 론

본 연구에서는 비교적 저렴한 비용의 장비로 누구나 쉽게 설치 및 운영할 수 있도록 Jetson Xavier Board 와 YOLOv5 객체 인식 기술을 활용한 차량 모니터링 시스템을 제안하였다. 시스템에는 OpenCV, PyTorch 등의 라이브러리가 활용되었고 프레임 간 동일 객체를 판단하고 추적하기 위한 객체 추적 알고리즘, 교통 정보를 추출하기 위한 차량 수 측정 알고리즘과 차량 속도 측정 알고리즘, Jetson Board에서 서버로 데이터를 전송하기 위한 TCP/IP Socket 통신, 시각화된 교통 정보를 제공하는 Web Server를 구현하였다.

개발된 차량 모니터링 시스템의 성능을 평가하고 시스템을 통해 얻은 교통 정보의 신뢰성을 검증하기 위해 진행한 테스트 결과 Jetson Xavier Board 환경에서 문제없이 프로그램이 동작함을 확인하였고, 통행량과 차량 속도 등의 교통 정보에 대한 신뢰도 또한 기존 시스템보다 나은 수준임을 확인하였다.

본 연구에서 제안한 차량 모니터링 시스템은 실험을 일몰 전까지 시행하였다. 객체 인식은 화면의 이미지를 활용하여 판단하기에 빛의 양이 적어지는 일몰 이후부터는 정확도가 낮아졌다. 향후 연구에서는 학습 단계에서 어두운 환경의 차량을 학습하여 적용한다면 정확도 높은 시스템이 될 것이다. 또한 교통 정보를 시각화한 그래프를 일일 단위로만 나타내는 것이 아닌 실시간으로 변화하도록 나타내고 차량 번호판 인식 알고리즘을 사용하여 과속한 차량의 번호판을 저장하는 기능을 추가한다면 더욱 활용도 높은 시스템이 될 것이다.

References

[1] D. W. Choi, "Active ITS infrastructure management strategy for enhanced ITS service," *J. KCA*, vol. 14, no. 9, pp. 45-53, Sep. 2014.

[2] Y. H. Lee and Y. S. Kim, “Comparison of CNN and YOLO for object detection,” *J. Semiconductor & Display Technol.*, vol. 19, no. 1, pp. 85-92, Mar. 2020.

[3] C. J. Lin, S. Y. Jeng, and H. W. Lioa, “A real-time vehicle counting, speed estimation, and classification system based on virtual detection zone and YOLO,” *Math. Problems in Eng.*, 2021.

[4] G. Jocher, *YOLOv5*(2020), Retrieved Aug. 13, 2021, from https://github.com/ultralytics/yolov5

[5] S. J. Yang, I. H. Jung, D. H. Kang, H. B. Baek, “Real-Time multi-object tracking using mixture of SORT and DeepSORT,” *J. KIIT*, vol. 19, no. 10, pp. 1-9, Oct. 2021.

[6] W. S. Choi, *DeepSort*(2021), Retrieved Oct. 01, 2021, from https://wansook0316.github.io/ds/dl/2021/03/14/computer-vision-17-DeepSort.html

[7] D. Ibrahim, *DeepSORT Algorithm*(2020), Retrieved Sep. 24, 2021, from https://github.com/mribrahim/yolov5-tracking

[8] J. T .Woo, *Concepts, Features and Differences of TCP/UDP*(2020), Retrieved Jan. 24. 2022, from https://coding-factory.tistory.com/614

[9] Intel, *OpenCV-Optical Flow*(2021), Retrieved Apr. 27, 2022, from https://docs.opencv.org/3.4/d4/dee/tutorial_optical_flow.html

[10] M. Jegal, *Speed camera enforcement standards, is it okay up to the speed limit of +10 km/h?*(2021), Retrieved Jan. 25, 2022, from https://www.sisaweek.com/news/articleView.html?idxno=143852

장 진 호 (Jin-ho Jang)

2022년 2월 : 부산대학교 정보컴퓨터공학부 학사

2022년 3월~현재 : 부산대학교 정보융합공학과 석사과정

<관심분야> 컴퓨터공학, 통신 공학, 네트워크

[ORCID:0000-0003-0382-2248]

박 은 영 (Eun-young Park)

2022년 2월 : 부산대학교 정보컴퓨터공학부 학사

2022년 3월~현재 : 부산대학교 정보융합공학과 석사과정

<관심분야> 컴퓨터공학, AI, 네트워크

[ORCID:0000-0002-7908-6059]

황 정 수 (Jeongsoo Hwang)

2022년 2월 : 부산대학교 정보컴퓨터공학부 학사

<관심분야> 소프트웨어공학, AI

[ORCID:0000-0002-9777-156X]

유 영 환 (Younghwan Yoo)

2004년 2월 : 서울대학교 전기컴퓨터공학부 박사

2007년 3월~현재 : 부산대학교 정보컴퓨터공학부 교수

<관심분야> 이동통신, 무선네트워크

[ORCID:0000-0002-2813-6116]