◎ 논 문

http://dx.doi.org/10.22680/kasa2022.14.4.006

저속 특장차의 도심 자율주행을 위한 신호등 인지 알고리즘 적용 및 검증

윤원섭* · 김종탁** · 이명규*** · 김원균****,†

Implementation and Validation of Traffic Light Recognition Algorithm for Low-speed Special Purpose Vehicles in an Urban Autonomous Environment

Wonsub Yun*, Jongtak Kim**, Myeonggyu Lee***, Wongun Kim****,†

Key Words: Low-speed unmanned special vehicle(저속 무인 특장차), Deep Learning(딥러닝), Traffic light recognition Algorithm(신호등 인지 알고리즘), Real-Time Object Detection(실시간 사물 감지), YOLO algorithm

ABSTRACT

In this study, a traffic light recognition algorithm was implemented and validated for low-speed special purpose vehicles in an urban environment. Real-time image data using a camera and YOLO algorithm were applied. Two methods were presented to increase the accuracy of the traffic light recognition algorithm, and it was confirmed that the second method had the higher accuracy according to the traffic light type. In addition, it was confirmed that the optimal YOLO algorithm was YOLO v5m, which has over 98% mAP values and higher efficiency. In the future, it is thought that the traffic light recognition algorithm can be used as a dual system to secure the platform safety in the traffic information error of C-ITS.

1. 서 론

자율주행 기술의 도래로 인해 다양한 산업군에 자율주행 플랫폼이 개발 및 적용되고 있으며, 자율주행 요소 기술인 인지, 판단 및 제어 기술은 인공지능 알고리즘 개발과 더불어 고도화가 되고 있다. 그러나 자율주행 기술은 시스템 오류 및 주변상황에 따른 심각한 물적 인적 피해를 유발할 수 있다. 이에 대응하기 위해 시스템의 Fail-Safe 기능을 인공지능 알고리즘 활용 기법, 시스템 이중화 기술 및 V2X(Vehicle-to-Everything) 활용을 통해 구현하고 있다.

대부분의 자율주행 기술은 Lidar, Radar 및 초음파 센서를 활용하여 주변 사물, 차선, 차량 등을 인지하고 주행에 필요한 기반 데이터를 활용하고 있다. 이와 더불어 인공지능 발전으로 인하여 카메라를 활용한 영상 데이터 인지 기술을 개발하고 있으며, 기존 센서데이터 시스템과 차량 외부로부터의 도로 신호 정보를 기반으로 2중화 설계를 도모하고 있다. 특히 도로 신호정보의 경우 V2X에서 획득한 정보와 영상기반의 신호정보를 활용하여 자율주행 차량의 안정성을 확보하고 있다.

특히 인공지능 알고리즘을 활용한 신호등 인지의 경우 도로의 규격 및 형태에 따라 신호등 종류 및 신호정보가 다르며, 자율주행 차량의 속도 및 형상에 따른 정형화된 기반데이터의 획득의 어려움이 있다. Saini et al.[(1)]은 HSV

* 한국생산기술연구원, 연구원
** 한국생산기술연구원, 선임연구원
*** 한국생산기술연구원, 연구원
**** 한국생산기술연구원, 수석연구원
†교신저자, E-mail: wgk@kitech.re.kr

자동차안전학회지: 제14권, 제4호, pp. 6~15, 2022
논문접수일: 2021.9.10, 논문수정일(1차: 2022.4.8, 2차: 2022.11.16), 게재확정일: 2022.11.16

채널 기반 color segmentation, Maximally Stable Extremal Region(MSER), Histogram of Oriented Gradients(HOG) features, support vector machine(SVM) 등의 기법을 사용하여 다양한 광도와 날씨 등의 환경 아래에서 획득된 이미지들을 통해 다른 환경에서도 균일하게 탐지가 가능한 신호등 탐지 CNN 알고리즘을 개발하였다. Ouyang et al.[2]은 자율주행 차량에 광각 카메라를 설치하여 신호등 영상 데이터를 수집 후, 모델의 복잡성을 줄이고 정확도를 향상시킨 알고리즘을 개발하여 이를 Nvidia Jetson TX1, TX2에 탑재하였다. 모델 개발에는 HSV, HSI 채널을 모델 학습시에 사용하였으며, SSD 기반 모델을 이용하여 학습을 진행하였다. 학습된 모델은 자율주행 차량과 버스에 장착하여 모델의 검증을 진행하였다. Kento et al.[3]은 초고속 카메라를 사용하여, 신호등의 LED가 깜빡이는 프레임에 대한 해결 방안을 제시하였다. 이는 band pass filter, binarization, buffering 등의 모듈을 사용하여 알고리즘이 개발되었고, 추후 신호 판별에는 SVM 기법이 활용되었다. Ruturaj et al.[4]은 5가지 신호에 대해서 Inception V2 R-CNN 모델을 전이학습하여 신호등을 탐지하는 알고리즘을 개발하였고, Mello et al.[5]은 인공적인 교통 상황 데이터를 생성하는 기법을 개발하여 부족한 데이터의 보충 및 데이터의 불균형 문제를 해소하였다.

기존에 이루어진 연구들의 한계로는, 신호등의 구조에 따른 클래스의 세분화된 분류가 이루어지지 않았다는 단점이 존재한다. 본 연구에서는 이를 극복하기 위하여 두가지의 세분화된 방법론을 제시하였으며, 대한민국 신호등의 고유한 특징(3구 및 4구의 가로형 신호등 형태)에 대응 가능하도록 데이터셋을 구축하여 모델의 학습 및 평가를 진행하였다. 기존 연구들의 한계점 및 이에 대해 본 연구에서 이루어진 개선사항은 Table 1과 같다.

Table 1 Drawbacks of conventional methods and its improvements made in this study

Drawbacks	Improvements made in this study
Dataset composed of vertical traffic lights (Not adequate for Korean traffic environment)	• Horizontal traffic light dataset constructed especially for Korean traffic environment
Detection model detecting single light, rather than the entire signal	• Entire signal detection • Training separate models according to the traffic light shape
Scarce number of classes (red, green, yellow)	• 3 classes for 3-holed traffic lights • 6 classes for 6-holed traffic lights • Can detect complex signals

본 연구는 광주 규제자유특구혁신사업의 선정지인 광주 평동산업2단지에서 자율주행 실증 시험을 수행하였으며, 저속 무인 특장차인 무인노면청소차의 자율주행에 필요한 맞춤형 신호등 인지 알고리즘을 개발하였다. 무인노면청소차는 광주광역시 광산구 장록동 일대에 위치하고 있는 국가산업단지로서 산단 주요 도로 폭은 35m 산단 통행도로 폭은 20m이며, 총 거리 4.8km를 왕복하게 된다. 자율주행 중 차량의 청소작업이 함께 수행된다. 무인노면청소차의 시스템 구성은 자율주행 플랫폼, 자율작업 시스템, 이를 통제하는 관제센터로 구성되어 있다. 무인노면청소차는 관제센터 C-ITS(Cooperative Intelligehnt Transport System)의 교통신호, 도로 정보를 통해 주행 시나리오를 구성하고 플랫폼에 장착된 센싱 시스템을 통해 확보한 실시간 도로 환경를 연계하여 자율주행을 실시하게 된다.

C-ITS의 교통 정보오류에서 플랫폼의 안전을 확보하기 위해 교통신호 이원화 시스템을 구성하고자 하였으며, 이에 카메라를 활용한 실시간 신호등 인지 알고리즘이 필수적으로 요구되었다. 실시간 인지에 필요한 인공지능 알고리즘은 YOLO 알고리즘을 활용하였다. YOLO 버전에 따라 실제 플랫폼에 적합한 알고리즘을 검증할 필요가 있으며, 시스템에 적용시 안정성을 확보하는 연구를 수행하였다.

2. 인지 알고리즘

2.1. YOLO 알고리즘

딥러닝 모델은 스스로 특징점들을 직접 학습하고 인간이 이해하지 못하는 특징점을 탐지할 수 있다. 그러나 이러한 특징점들을 탐지하기 위해선 딥러닝 모델의 학습이 선행되어야 한다. 데이터 분석에서는 학습 용이성, 학습

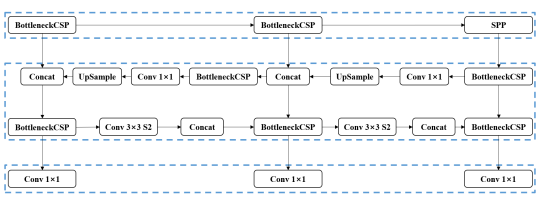


Fig. 1 Model architecture

Table 2 Model details

Features	• Binary classification using sigmoid for each class • Residual values transmitted using skip connection • Classification results outputted through Softmax	• Increased input resolution of the model • Increased number of layers to increase the receptive field • Reduced inference cost and memory cost	• Models are divided based on the depth of the layers • Sets the anchor box and uses it to create bounding box
mAP (%)	60.6	64.4	s – 55.4 m – 63.1 l – 66.9 x – 68.8

시간 단축, 신경망의 성능 향상과 같은 장점으로 인해 모델의 전이학습 방식이 채택되었다. 데이터 분석에 사용된 모델은 YOLO v3,[6] YOLO v4[7] 및 YOLO v5[8]이다. 사용된 모델의 구조 및 특징은 Table 2와 Fig. 1과 같다.

2.2. 모델 최적화

모델 선택 외에도 모델 최적화가 수행되었다. 모델의 일반적인 동작을 제어하는 매개변수들을 조정하여 딥러닝 모델의 최적화를 진행하였다. 매개변수 최적화를 위해 널리 채택된 세가지 방법[9]은 (1) manual search, (2) grid searches 및 (3) arbitrary search이다. 딥러닝 네트워크 학습과정 중에서 알고리즘을 이해하고 시각화하기 위한 여러 시도[10,11]가 있었지만, empirical approach[12]는 여전히 매개변수들을 최적화하는 가장 효과적인 접근 방식 중 하나이므로 데이터 분석에 적용되었다.

3. 데이터 획득

3.1. 데이터 획득 방법

두 가지 방법의 데이터 분석 방법은 Fig. 2와 같다. 본 연구에서는 두가지 방법론으로 데이터 획득을 수행하였다. Fig. 2에 표시된 Method 1은 100% 영상 기반 알고리즘을 만드는 것을 목적으로 신호등 한 개의 클래스에 대해서 탐지 알고리즘을 학습 후, 탐지 알고리즘에서 획득된 신호등의 영역을 추출한다. 이를 바탕으로 신호 판별 알고리즘을 학습하여, 신호등의 신호를 분류 및 판별을 진행하였다. Fig. 2에 표시된 Method 2는 신호등의 신호 기반 라벨링을 진행한 후, 이를 신호등의 형태(3구, 4구)에 따라 데이터셋을 분류하였다. 분류된 데이터셋에 따라 각각 탐지 알고리즘을 학습 후, 신호등의 종류에 따라 학습된 가중치를 전환하여 탐지 및 분류를 진행하였다.

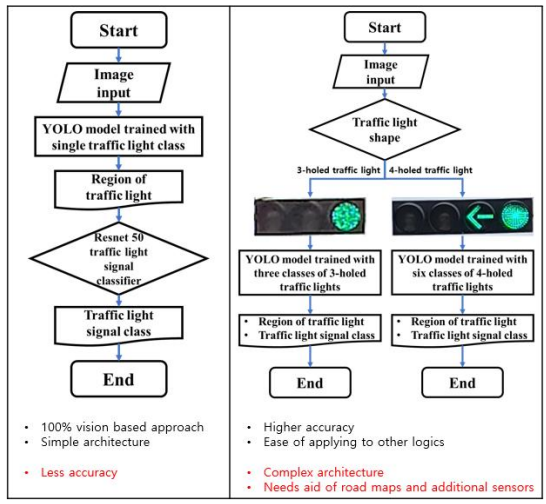


Fig. 2 Experiment flowchart

3.2. 플랫폼 구성 및 데이터 처리 시스템

기반 데이터 확보를 위해 Leopard Imaging 사의 LI-IMX390-GMSL2-060H 카메라를 사용하였으며, Fig. 3와 같이 무인 노면청소차의 전면 상단에 장착하였다. 플랫

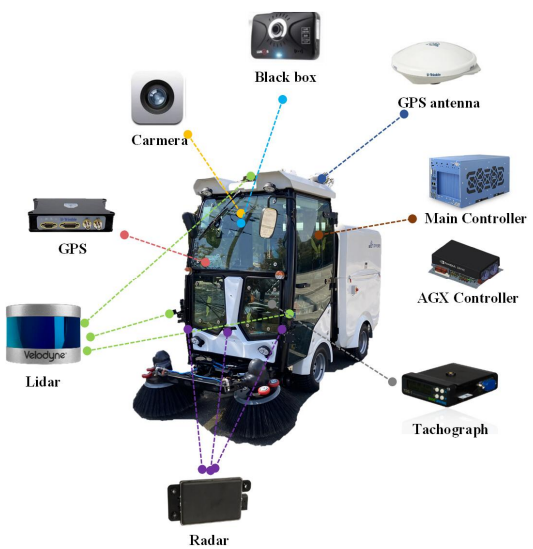


Fig. 3 Device overview

폼에 적용된 카메라는 1937 × 1217 크기의 화각, 30fps의 데이터 처리 속도, 파크라 기반 데이터 전송 등의 이점이 있어 신호등 데이터 획득 및 실시간 신호등 탐지에 활용하였다. 카메라는 무인노면청소차에 장착되어 광주 평동산업 2단지 4.8km 구간에서 오전 10시부터 오후 4시까지 영상 정보를 확보하였다. 자세한 경로는 Fig. 4에 나타내었다. 상기 구간에서 알고리즘이 인지해야 되는 신호등의 정보는 Table 3과 같다.

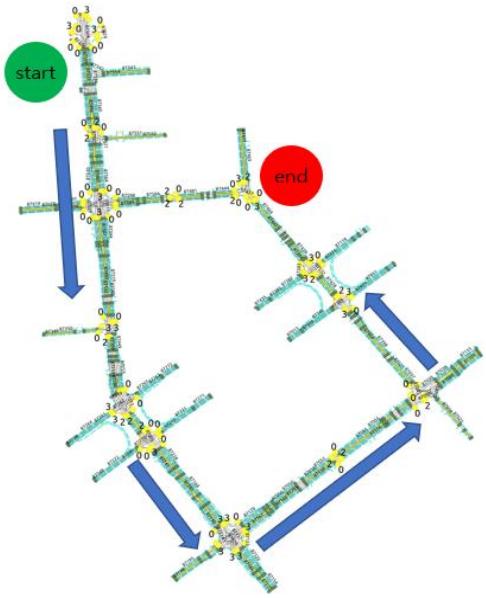


Fig. 4 Map data of our experiment. 0: Passenger crossing signal, 2: 3 holed traffic light, 3: 4 holed traffic light

Table 3 Sets and numbers of traffic lights in our experiment path

Type	#sets	#traffic lights
3 holed traffic lights	18	32
4 holed traffic lights	18	45

Table 4 Dataset details

Hardware / Software specification	
CPU	Intel® Xeon® E-2278G
Graphic Cards (GPU)	Tesla T4 × 2
RAM	64 GB
Operating System	Ubuntu 18.04.3 LTS
Graphic Drivers	CUDA 11.2 & cuDNN8.1.1
Python Version	3.6.8

본 연구에서 수행된 데이터 분석은 Darknet 라이브러리를 기반으로 하였으며, CUDA 11.2 및 cuDNN 8.1.1 그래픽 드라이버 환경에서 두 개의 Tesla T4 GPU를 사용하였다. 64 GB의 RAM과 Intel® Xeon® E-2278G CPU를 사용하여 GPU의 성능을 지원하였다. 데이터 분석 장치의 사양은 Table 4와 같다.

3.3. 신호등 인지 최소 요구거리

신호등 신호인지는 무인노면청소차의 최단 정지거리 이전에서 필수적으로 판별되어야 한다. 또한 무인노면청소차의 최단 정지거리는 비상정지 상황에서 발생하는 정지거리이다. 이에 따라 신호등 인지 필수거리는 무인노면청소차의 비상정지에서 필요한 최단거리를 기준으로 결정되며, 필수거리에서 시험 데이터를 확보하여야 한다.

무인노면청소차는 비상정지 상황은 관제센터의 통제에 따르며, 관제센터의 비상정지 신호가 통신망에 의해 무인노면청소차의 브레이크 엑츄에이터를 동작시킨다. 통신 딜레이 시간은 0.103s이며, 이후 엑츄에이터(공압 밸브) 동작에 의해 감속이 발생한다. 감속으로 차량 정지까지 소요되는 시간은 0.7초이며, 감가속도가 일정하다고 가정할 때 6km/h에서 정지까지 0.7초의 감가속도는 2.38m/s^{2}이다. 따라서 무인노면청소차의 비상정지 거리는 다음과 같다.

$$s = v_o t_1 + v_o t_2 - \frac{1}{2} a t_2^2 \tag{1}$$

$$v_o = 1.667 m/s \tag{2}$$

$$a = \frac{v_0}{t_2} = \frac{1.667 m/s}{0.7 s} = 2.38 m/s^2 \tag{3}$$

$$\therefore s = 0.755 m \tag{4}$$

위 식에서 v_0는 무인노면청소차의 평균주행속도이며, t_1와 t_2는 통신 딜레이 시간 및 감속으로 차량 정지까지 소요된 시간이며, a는 무인노면청소차의 비상정지시 감가속도이다.

따라서, 비상정지 신호에 의해 차량이 완전정지되는 총 0.803s 동안 무인노면청소차가 이동한 정지거리는 0.755m이다. 따라서 계산된 0.755m 전후로 신호등 영상을 촬영하였으며, 획득한 파크라 기반 byte-array 신호를 usb 모듈로 전송하여 이를 opencv 라이브러리가 인식 가능한 RGB 배열로 변환하여 데이터를 처리하였다.

3.4. 데이터셋 가공

데이터셋 가공을 위해 총 1220개의 영상 데이터가 활용되었으며, 248개의 신호등 탐지용 데이터, 764개의 4구 신호등 데이터, 208개의 3구 신호등 데이터로 분류하였다. 신호등 탐지용 데이터에는 신호등 하나의 클래스를 기반으로, 174개의 영상 데이터가 학습 데이터셋, 74개의 영상 데이터가 검증 데이터셋으로 활용되었다. 신호등 데이터셋의 정보는 Table 5와 같다. 3구 신호등 알고리즘 학습에는 청신호, 주황신호, 적신호가 포함된 3개의 클래스를 기반으로, 146개의 영상 데이터가 학습 데이터셋, 62개의 영상 데이터가 검증 데이터셋으로 활용되었다. 3구 신호등 데이터셋 및 클래스명에 대한 정보는 Table 6, Table 7과 같다. 4구 신호등 알고리즘 학습에는 적신호, 주황신호, 적신호, 직진-좌회전 신호, 정지-좌회전 신호, 정지-주황 신호가 포함된 6개의 클래스를 기반으로 535개의 영상 데이터가 학습 데이터셋, 229개의 영상 데이터가 검증 데이터셋으로 활용되었다. 4구 신호등 데이터셋 및 클래스명에 대한 정보는 Table 8, Table 9와 같다.

Table 5 Method 1 (separate detection and classification) data details

Dataset	Class	Training		Testing	
		Images	#Traffic lights	Images	#Traffic lights
Traffic lights	TL	174	407	74	175

Table 6 Method 2 (3 holed traffic lights) data details

Dataset	Class	Training		Testing	
		Images	#Traffic lights	Images	#Traffic lights
3-Holed Traffic lights	BBG	106	227	45	97
	BYB	10	20	4	8
	RBB	30	77	13	33

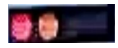
Table 7 Method 2 (3 holed traffic lights) class details

Class	Full name	Image
BBG	Green light	
BYB	Yellow light	
RBB	Red light	

Table 8 Method 2 (4 holed traffic lights) data details

Dataset	Class	Training		Testing	
		Images	#Traffic lights	Images	#Traffic lights
4-Holed Traffic lights	RBBB	42	92	18	39
	BYBB	54	133	23	57
	RYBB	86	172	47	74
	BBBG	57	115	24	49
	RBGB	27	54	11	23
	BBGG	13	29	6	12

Table 9 Method 2 (4 holed traffic lights) class details

Class	Full name	Image
RBBB	Straight stop	
BYBB	Straight proceed with caution	
RYBB	Straight stop, left turn proceed with caution	
BBBG	Straight proceed	
RBGB	Straight stop, left turn proceed	
BBGG	Straight proceed, left turn proceed	

Python의 labelImg[13] 라이브러리를 사용하여 이미지의 각 신호등에 대해 라벨링 작업을 진행하였다. 라벨링을 수행하는 동안 라벨링 bounding box가 신호등을 촘촘하게 감싸게 함으로써, 라벨링 상태에 따른 정확도를 확보하였다. 라벨링 데이터는 bounding box의 x 및 y 좌표 및 클래스가 포함된 txt 파일 형식으로 저장되었다.

4. 결과 및 고찰

4.1. 인지 성능 평가

본 연구에서 신호등 탐지 모델을 평가하기 위해 *precision*, *recall*, *F1-score* 및 *mean average precision(mAP)*을 적용하여 성능 평가를 수행하였으며, 이에 관련된 식은 다음과 같다.

$$precision(Pre) = \frac{True\ Positive}{True\ Positive + False\ Positive} \quad (5)$$

$$recall\,(Rec) = \frac{True\,Positive}{True\,Positive + False\,Negative} \tag{6}$$

$$F_1 - score = 2 \times \frac{Pre \times Rec}{Pre + Rec} \tag{7}$$

$$mAP = \frac{1}{Num(K)} \sum_{x \in K} AP_x \tag{8}$$

여기서 *True Positive*는 정확하게 탐지된 신호등의 수, *False Positive*는 오인된 신호등의 수, *False Negative*는 탐지되지 않은 신호등의 수, *K*는 IOU기반 임계값들의 집합, *Num(K)*는 집합 *K*의 원소의 수, *APx*는 x보다 큰 모든 *recall* 값의 최대 정밀도이다.

만약 *precision*, *recall*, *F1−score* 및 *mAP*가 1에 가까우면 학습된 모델이 더 높은 탐지 정확도를 보여주고, 값이 0에 가까우면 학습된 모델의 탐지 정확도가 낮아진다.

4.2. 신호등 인지 성능

학습된 모델의 시각화를 통해 모델의 검증을 수행하였다. 학습이 진행됨에 따라 모델은 대상 후보군에 표시되는 수많은 앵커를 생성하였으며, 각 컨볼루션 층의 가중치를 추출하여 추가적으로 시각화를 진행하였다. 신호등 탐지 모델의 최종 아키텍처를 백본 및 다양한 매개변수로 구성된 다수의 조합을 검증하여 선택하였다. 초기 epoch는 250으로 설정되었으며 최상의 성능 지점을 얻기 위해 모

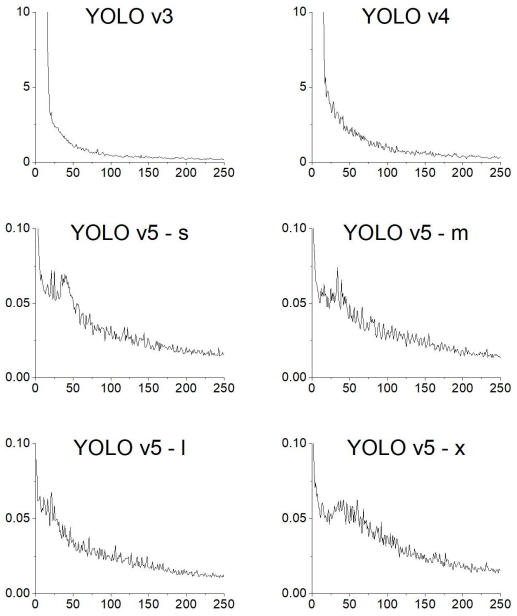


Fig. 6 Training loss of six different YOLO models trained with three holed traffic lights dataset

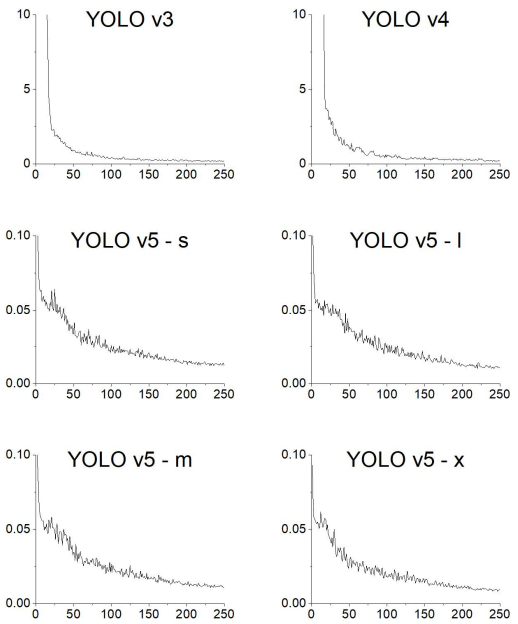


Fig. 5 Training loss of six different YOLO models trained with one classed traffic light dataset

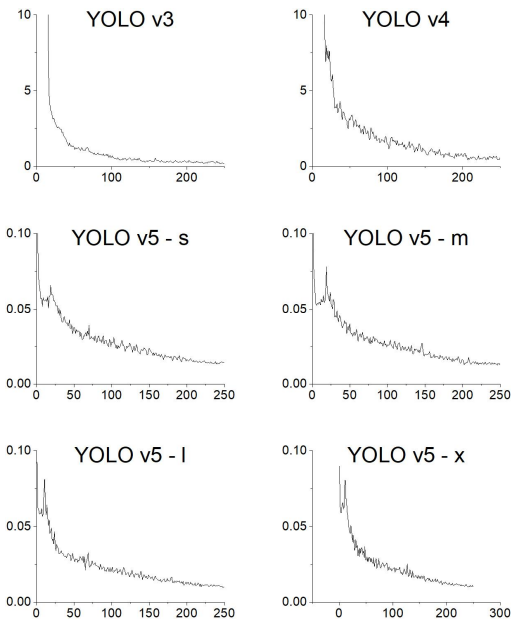


Fig. 7 Training loss of six different YOLO models trained with four holed traffic lights dataset

든 epoch에 대해 체크 포인트를 저장하였다. 최적의 매개변수는 learning rate 0.001, momentum 0.9, batch 크기 3으로 결정되었다. 3개의 데이터셋에 대한 YOLO 모델의 학습 중 손실률은 Fig. 5, Fig. 6, Fig. 7과 같다.

학습이 진행됨에 따라 훈련 정확도가 지속적으로 증가되었으며, 학습된 모델의 false positive rate은 recall과 관련하여 일정한 비율 감소를 확인되었다. 모델 성능은 검증 데이터셋으로 검증되었다.

4.3. Method 1: 신호등 탐지 결과

Fig. 1에 명시된 Method 1을 토대로, ResNet 50 모델과 획득한 신호등의 영역 영상을 사용하여 신호등 판별 모델을 학습하였다. 초기 epoch는 100으로 설정하였으며, learning rate 0.001, momentum 0.9, batch size 4로 매개변수를 설정하였다. 판별 모델의 가중치는 가장 높은 정확도를 보이는 epoch의 가중치가 저장되었다.

신호등 1개의 클래스에 대한 탐지 모델의 경우, Table 10과 같이, YOLO v5x 모델이 가장 높은 0.97의 *F1-score*을 확인할 수 있었으나, YOLO v4 모델이 가장 높은 98.87%의 *mAP* 값을 보여주었다. Table 11과 같이, 클래스별 정확도는 직진, 직진-좌회전, 정지, 대기 신호가 80% 이상

Table 10 Performance of YOLO models in single class traffic light detection

Dataset	Model	Precision	Recall	F1-Score	mAP (%)
Traffic lights	v3	0.94	0.97	0.95	98.38
	v4	0.92	0.97	0.94	98.87
	v5s	0.93	0.99	0.96	98.03
	v5m	0.94	0.98	0.96	97.6
	v5l	0.93	0.99	0.96	97.98
	v5x	0.94	0.99	0.97	98.18

Table 11 Performance of trained classification model on various classes

Class	Accuracy (%)
Green	83.30
Green-Left	88.90
Red	91.67
Red-Left	76.92
Red-Yellow	36.33
Yellow	100.00

Fig. 8 Method 1 (separate detection and classification) results

의 *mAP*를 나타내는데 반해, 정지-좌회전 및 정지-대기 신호는 각각 76.92%, 36.33%의 *mAP*를 나타내었다. 학습된 모델의 결과는 Fig. 8과 같다.

4.4. Method 2: 신호등 탐지 결과

4.4.1. 3구 신호등

Method 2의 3구 신호등 탐지 및 분류 모델의 경우, Table 12와 같이, YOLO v5m 모델이 가장 높은 0.96의 *F1-score* 및 98.4%의 *mAP* 값을 보여주었다. 따라서 3구 신호등 탐지 및 분류 모델로는 YOLO v5m 모델이 선정되었다. Table 13에서 확인할 수 있듯이 클래스별 *mAP*

Table 12 Performance of YOLO models in 3-holed traffic lights detection

Dataset	Model	Precision	Recall	F1-Score	mAP (%)
3-Holed Traffic lights	v3	0.97	0.94	0.95	97.03
	v4	0.85	0.95	0.9	97.13
	v5s	0.96	0.85	0.9	91.06
	v5m	0.92	1.00	0.96	98.4
	v5l	0.89	0.95	0.92	95.14
	v5x	0.98	0.97	0.98	98.25

Table 13 Performance of 3-holed traffic lights detection model on various classes

Dataset	Model	Class	mAP (%)
3-Holed Traffic lights	YOLO v5m	BBG	98.40
		BYB	99.50
		RBB	97.30

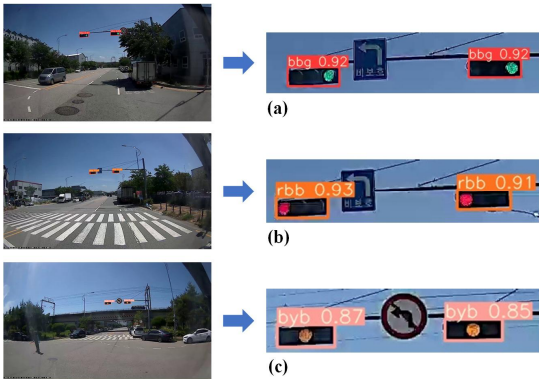


Fig. 9 Method 2 (3 holed traffic lights) detection results. (a) BBG (green light) detection (b) BYB (yellow light) detection (c) BBR (red light) detection

값은 모두 97% 이상의 값을 나타내으며, 학습된 모델의 정확도를 검증할 수 있었다. 학습된 모델의 결과는 Fig. 9와 같다.

4.4.2. 4구 신호등

Method 2의 4구 신호등 탐지 및 분류 모델의 경우, Table 14와 같이, YOLO v5m 이 가장 높은 0.97의 *F1-score*을 보인 데 반해, YOLO v3의 모델이 가장 높은 99.67%의 *mAP* 값을 보여주었다. Table 15와 같이 각 클래스별 정확도를 나타냈으며, 모두 98% 이상의 값을 나타내었다. 학습된 모델의 결과는 Fig. 10과 같다.

Table 14 Performance of YOLO models in 4-holed traffic lights detection

Dataset	Model	Precision	Recall	F1-Score	mAP (%)
4-Holed Traffic lights	v3	0.96	0.97	0.96	99.67
	v4	0.88	0.99	0.93	97.93
	v5s	0.95	0.95	0.95	98.58
	v5m	0.98	0.95	0.97	98.94
	v5l	0.95	0.97	0.96	98.86
	v5x	0.97	0.95	0.96	98.85

Table 15 Performance of 4-holed traffic lights detection model on various classes

Dataset	Model	Class	mAP (%)
4-Holed Traffic lights	YOLO v5m	RBBB	97.60
		BYBB	91.60
		RYBB	99.50
		BBBG	99.50
		RBGB	93.20
		BBGG	99.70

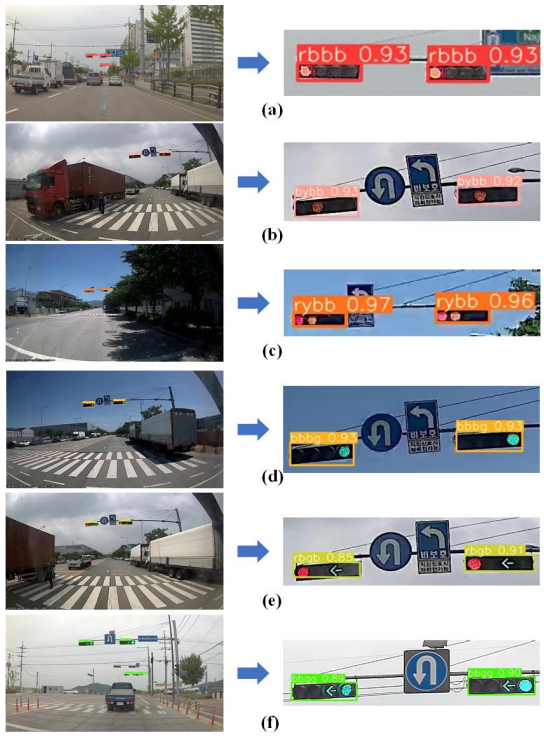


Fig. 10 Method 2 (4 holed traffic lights) detection results. (a) RBBB (red light) detection (b) BYBB (yellow light) detection (c) RYBB (red light and yellow light) detection (d) BBBG (green light) detection (e) RBGB (red light and left sign) detection (f) BBGG (green light and left sign) detection

4.4.3. YOLO 알고리즘 성능 평가

학습된 모델을 무인 노면 청소차량에 적용하였을 때, 이를 최적화하기 위하여 모델의 GPU 및 CPU 점유율, *mAP*, 탐지 소요 시간을 변수로 적용하여 다음의 공식을 도출하였다.

$$Efficiency = \frac{6}{CPU_{\min} + CPU_{\max}} + \frac{250}{GPU} + 0.25 \times mAP + \frac{0.00125}{t} \quad (8)$$

여기서 CPU_{min} 및 CPU_{max}는 CPU의 최소, 최대 점유율, GPU는 GPU의 점유율, mAP는 전이학습이 이루어지긴 전 모델의 정밀도, t는 프레임당 처리속도를 나타낸다.

학습된 모델의 프레임당 탐지 소요 시간은 Table 16에 나타내었으며, CPU 및 GPU 점유율은 Table 17에 기술하였다. YOLO v3와 YOLO v4 모델은 다른 모델들에 비해 높은 CPU 점유율은 보인데 반해, GPU 점유율은 상대적으로 낮았다. YOLO v5 모델들은 학습된 가중치의 크기에 따라 GPU 점유율이 상이하였고, CPU 사용률은 YOLO v3, YOLO v4 모델에 비해 낮음을 확인하였다. 따라서 YOLO 버전에 따른 종합적 알고리즘 효과 분석을 위해 식 (8)를 적용하였으며, 모델별 효과성 계수는 Table 18에 나타내었다. 분석 결과에서 효과성 계수는 YOLO v3 가 0.71로 가장 높았으며, YOLO v5x가 0.5로 가장 낮게 평가되었다. 이는 YOLO v5x는 CPU의 사용량에 비해 비해 GPU의 사용량이 많으며, 이로 인하여 플랫폼 시스템의 성능에 의존적이라고 할 수 있다. 무인 저속 특장차의 시스템 특성상 0.55 이상의 *Efficiency*를 나타낼 경우, 주행 중 실시간 영상으로 신호등을 탐지하는데 하드웨어적으로 문제가 발생하지 않을 것으로 판단된다. 따라서 0.56의 *Efficiency*를 보유하고, 98.4%(3구) 및 98.94%(4구)의 *mAP*를 나타낸 YOLO v5m 모델이 현장 사용에 효율적이라고 판단된다.

Table 16 Evaluation duration of various models

Model (YOLO)	Evaluation time (sec/frame)
v3	0.006
v4	0.013
v5s	0.010
v5m	0.025
v5l	0.050
v5x	0.089

Table 17 CPU and GPU usage of the trained models

Model (YOLO)	CPU usage (%)		GPU usage (MB)
	Min	Max	
v3	12.3	27.1	2429
v4	11.0	26.0	3339
v5s	5.5	19.6	2653
v5m	6.0	20.2	6221
v5l	6.3	21.2	9853
v5x	6.9	20.9	8489

Table 18 Efficiency factor of the trained models

Dataset	Model	Efficiency
Traffic lights	v3	0.71
	v4	0.58
	v5s	0.70
	v5m	0.56
	v5l	0.51
	v5x	0.50

4.4.4. 고찰

획득한 신호등 데이터를 학습시킨 결과, Fig. 2에서 소개된 두가지 방법 중, ResNet 50 기반 판별 알고리즘인 Method 1 보다 YOLO 기반 탐지 및 분류 알고리즘인 Method 2의 정확도가 높음을 확인하였으며, Method 2가 신호등 탐지 및 분류 상황에서 더욱 유리하게 작용할 것으로 예상된다. 또한 데이터 분석이 진행된 Tesla T4 GPU에서 YOLO v5l 모델까지는 영상 프레임이 저하되는 현상이 발생하지 않았지만, GPU 메모리의 한계인 16 GB를 다 사용하는 YOLO v5x 모델의 경우, 탐지 알고리즘이 GPU 메모리를 확보하기 위하여 프레임을 저하하는 현상이 발생하였다. 하지만 신호등 탐지에서 YOLO V5x와 같이 구조성이 복잡한 모델이 현재 신호등 분류 과제에서 큰 성능을 발휘하지 못한다는 것은 본 연구에서의 데이터 분석으로 입증되었다. 따라서 무인 저속 특장차의 특성상, 우수한 *mAP* 및 *Efficiency*를 지닌 YOLO v5m 모델을 사용하는 것이 효율적이라고 판단된다.

5. 결 론

본 연구에서는 저속 특장차인 무인노면청소차에 적합한 신호등 인지 알고리즘 개발을 수행하였으며, YOLO 버전에 따른 탐지 효율을 검증하였다. 연구의 결론은 다음과 같다.

1) Method 1 기반 신호등 1개의 클래스에 대한 탐지 모델의 경우, YOLO v5x 모델이 가장 높은 *F1-score*를 확인하였으나, 평균 정확도는 YOLO v4 모델이 가장 높은 것으로 나타났다. 그러나 특정 신호에 대해 매우 부정확한 결과를 보임으로써 Method 1은 무인 저속 특장차의 신호등 인지 알고리즘으로 부

적절함을 확인할 수 있었다.

2) 신호등의 형태(3구, 4구)에 따라 데이터셋을 분류한 Method 2은 Method 1에 비해 교통 신호 클래스에 대해 평균적으로 높은 정확도를 보였으며, Method 1에서 취약했던 주황색 신호 탐지도 매우 높은 정확도를 확보할 수 있었다.

3) 결과적으로 신호등 인지 알고리즘의 정확도를 높이기 위해 Method 1 및 Method 2의 효과성을 평가한 결과 Method 2의 방법이 무인 저속 특장차에 효과적임을 확인하였으며, YOLO v5m 모델이 가장 적합함을 확인하였다.

4) 본 연구에서 확보한 실시간 신호등 인지 알고리즘은 C-ITS의 교통 정보오류에서 플랫폼의 안전을 확보하기 위한 교통신호 이원화 시스템으로써 활용 가능할 것으로 판단된다, 향후 C-ITS와 연계한 고신뢰성 신호인지 알고리즘을 개발하여 무인 저속 특장차의 신호등 인지 성능 개선에 사용할 예정이다.

후 기

본 논문은 중소벤처기업부 '규제자유특구혁신사업육성(R&D)'의 지원을 받아 연구되었음(No. P0012726).

참고문헌

(1) S. Saini, S. Nikhil, K. R. Konda, H. S. Bharadwaj, and N. Ganeshan, 2017, "An efficient vision-based traffic light detection and state recognition for autonomous vehicles", in *2017 IEEE Intelligent Vehicles Symposium (IV)*, pp. 606~611.

(2) Z. Ouyang, J. Niu, Y. Liu, and M. Guizani, 2019, "Deep CNN-based real-time traffic light detector for self-driving vehicles", *IEEE transactions on Mobile Computing*, Vol. 19, No. 2, pp. 300~313.

(3) K. Yabuuchi, M. Hirano, T. Senoo, N. Kishi, and M. Ishikawa, 2020, "Real-time traffic light detection with frequency patterns using a high-speed camera", *Sensors*, Vol. 20, No. 14, p. 4035.

(4) Kulkarni, Ruturaj, Shruti Dhavalikar, and Sonal Bangar, "Traffic light detection and recognition for self driving cars using deep learning", 2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA), IEEE, 2018.

(5) de Mello, Jean Pablo Vieira, et al., "Deep traffic light detection by overlaying synthetic context on arbitrary natural images", Computers & Graphics 94(2021): 76~86.

(6) J. Redmon and A. Farhadi, 2018, "Yolov3: An incremental improvement", arXiv preprint arXiv: 1804.02767.

(7) A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, 2020, "Yolov4: Optimal speed and accuracy of object detection", arXiv preprint arXiv:2004.10934.

(8) A. S. Glenn Jocher, Jirka Borovec, NanoCode012, Ayush Chaurasia, TaoXie, Liu Changyu, Abhiram V, Laughing, tkianai, yxNONG, Adam Hogan, lorenzomammana, AlexWang1900, Jan Hajek, Laurentiu Diaconu, Marc, Yonghye Kwon, oleg, wanghaoyang 0106, Yann Defretin, Aditya Lohia, ml5ah, Ben Milanko, Benjamin Fineran, Daniel Khromov, Ding Yiwei, Doug, Durgesh, and Francisco Ingham, 2021, "ultralytics/yolov5: v5.0 - YOLOv5-P6 1280 models, AWS, Supervise.ly and YouTube integrations", ed. Zenodo.

(9) S. R. Young, D. C. Rose, T. P. Karnowski, S.-H. Lim, and R. M. Patton, 2015, "Optimizing deep learning hyper-parameters through an evolutionary algorithm", in Proceedings of the Workshop on Machine Learning in High-Performance Computing Environments, pp. 1~5.

(10) H. Liu, T. Taniguchi, Y. Tanaka, K. Takenaka, and T. Bando, 2017, "Visualization of driving behavior based on hidden feature extraction by using deep learning", IEEE Transactions on Intelligent Transportation Systems, vol. 18, no. 9, pp. 2477~2489.

(11) J. Yosinski, J. Clune, A. Nguyen, T. Fuchs, and H. Lipson, 2015, "Understanding neural networks through deep visualization", arXiv preprint arXiv: 1506.06579.

(12) J. Han, J. Pei, and M. Kamber, 2011, Data mining: concepts and techniques. Elsevier.

(13) Tzutalin. LabelImg [Online] Available: https://github.com/tzutalin/labelImg