◎ 논 문

http://dx.doi.org/10.22680/kasa2019.11.2.035

# 자율주행 제어를 위한 향상된 주변환경 인식 알고리즘

배인환* · 김영후* · 김태경* · 오민호* · 주현수* · 김슬기* · 신관준* ·
윤선재* · 이채진* · 임용섭* · 최경호*

# Improved Environment Recognition Algorithms for Autonomous Vehicle Control

Inhwan Bae*, Yeounghoo Kim*, Taekyung Kim*, Minho Oh*, Hyunsu Ju*,
Seulki Kim*, Gwanjun Shin*, Sunjae Yoon*, Chaejin Lee*,
Yongseob Lim*, Gyeungho Choi*

*Key Words : Autonomous vehicle(자율주행차), Cross-checking system(상호 확인 시스템), Image machine learning(이미지 기계 학습), Integrated control algorithm(통합제어 알고리즘), Lane detection(차선 인식), Obstacle detection and avoidance(물체 인식 및 회피), Recognition algorithm(인식 알고리즘), Sign detection(표지판 인식)*

ABSTRACT

This paper describes the improved environment recognition algorithms using some type of sensors like LiDAR and cameras. Additionally, integrated control algorithm for an autonomous vehicle is included. The integrated algorithm was based on C++ environment and supported the stability of the whole driving control algorithms. As to the improved vision algorithms, lane tracing and traffic sign recognition were mainly operated with three cameras. There are two algorithms developed for lane tracing, Improved Lane Tracing (ILT) and Histogram Extension (HIX). Two independent algorithms were combined into one algorithm - Enhanced Lane Tracing with Histogram Extension (ELIX). As for the enhanced traffic sign recognition algorithm, integrated Mutual Validation Procedure (MVP) by using three algorithms – Cascade, Reinforced DSIFT SVM and YOLO was developed. Comparing to the results for those, it is convincing that the precision of traffic sign recognition is substantially increased. With the LiDAR sensor, static and dynamic obstacle detection and obstacle avoidance algorithms were focused. Therefore, improved environment recognition algorithms, which are higher accuracy and faster processing speed than ones of the previous algorithms, were proposed. Moreover, by optimizing with integrated control algorithm, the memory issue of irregular system shutdown was prevented. Therefore, the maneuvering stability of the autonomous vehicle in severe environment were enhanced.

## 1. 서 론

현대 생활에서 자동차는 매우 보편적인 이동수단이다. 국토교통부가 2017년 12월에 제시한 자료에 따르면, 우리나라는 누적 자동차 등록대수가 2253만 대로 2.3인 당 1대 꼴로 자동차를 보유하였다. 뿐만 아니라, 신규 자동차 등록도 몇 년 간 꾸준히 증가하는 추세이다. 이는 자동차의 사용이 앞으로도 많아질 것임을 의미한다. 늘어난 자동차 수로 인해 도로 상황은 앞으로 더욱 혼잡지리라 예

* 대구경북과학기술원(DGIST) 융복합대학
E-mail : yslim73@dgist.ac.kr

상된다. 게다가 도로에는 자동차들만 있는 것이 아니다. 보행자, 자전거, 오토바이 등 다양한 요소들이 혼재하기 때문에 어떤 사고가 발생할 지 알 수 없다. 이와 같은 혼잡한 도로에서의 위험에 대처하고자 Google, Tesla 등 여러 회사들은 자체적으로 자율주행 기술 개발을 진행하고 있다.(1)

자율주행기술은 다양한 센서를 활용하여 주변환경을 인지한 후, 판단하고 반응하는 일련의 기술을 일컫는다. 이러한 기술이 발전하고 대중화된다면 우리의 삶을 안전하고 편안하게 만들어 줄 것이다. 하지만, 기술이나 센서의 한계로 인한 미판 혹은 오판 때문에 심각한 교통사고를 유발할 수 있다. 따라서 자율주행 기술을 더욱 신뢰할 수 있게끔 발전시킬 필요가 있다.(2)

본 논문에서는 라이다(Lidar)와 카메라(Camera) 두 종류의 센서를 활용하는 향상된 주변 환경 인식 알고리즘을 설계 및 개발하였고, 통합 제어 알고리즘을 체계화하였다. 먼저, 차선 인식을 개선하기 위하여, Enhanced Lane Tracing with Histogram Extension(ELIX)를 개발 및 검증하였다. 또한, 향상된 표지판 인식을 위하여, Mutual Validation Procedure(MVP)를 제안하였다. 위 두 가지 알고리즘을 통해 주변 환경에 대한 오판과 미판을 방지하고 더욱 정확하고 빠른 판단을 할 수 있도록 초점을 맞추었다. 위 모든 알고리즘을 수행하기 위해, 통합 제어 알고리즘 “혜안”을 설계하여 전체 구조를 잡았다.

## 2. 실험장치: 센서 및 하드웨어

### 2.1. 센서

자율주행 알고리즘을 테스트할 하드웨어 기반을 제작하는 데에 사용된 센서들의 재원은 Table 1과 같다. 비전(Vision) 센서는 차선, 표지판, 보조적으로 주변환경 인식을 목적으로 사용하였으며, 전방의 물체를 인식하기 위해서 2D LiDAR를 이용하였다.

Table 1 Specifications of the sensors

| Sort | Type | Spec. |
|---|---|---|
| LiDAR | LMS 151<br>(2D LiDAR) | Recognize up to 50m<br>Resolution up to 0.25°<br>Scan speed up to 50Hz |
| Vision | Logitech c930e<br>(Webcam) | 90° diagonal Field of View<br>Support up to 1080p 30fps |

### 2.2. 플랫폼

자율주행 알고리즘을 검증하기 위해 Unmanned Solution 사의 자율주행 로봇 플랫폼 ERP-42 를 사용하여 Fig. 1과 같이 제작하였다. 알루미늄 프로파일 구조물을 만들고, 선정된 비전 센서 3개와 LiDAR 센서 1개를 부착하여 고정하였다.

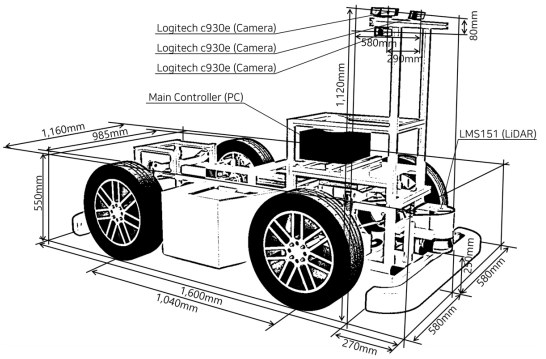



Fig. 1 The experimental platform for the autonomous system

자율주행 시스템이 작동할 주행 제어기의 경우, 다양한 센서로부터 입력되는 데이터를 실시간으로 안전하게 처리할 수 있도록 Table 2의 재원으로 제작하였다.

Table 2 Detail components of the controller

| Sort | Spec. |
|---|---|
| Mainboard | Advantech AIMB-275 001AE |
| CPU | Intel i7-6700 |
| GPU | Nvidia GTX 1050Ti LP |
| RAM | DDR4 19200 16G |

높은 노면 온도 위에서도 장시간 작동 및 주행하기 위한 안정성 테스트 또한 진행하였다. 주행 제어기 내부 공

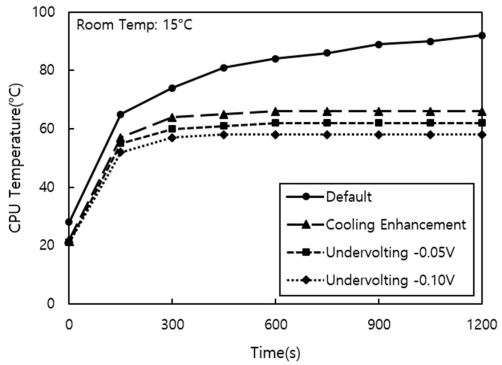



Fig. 2 Temperature change before/after control PC tuning

기 흐름을 개선하기 위해 부품들을 추가하였고, CPU의 언더볼팅(Undervolting)을 수행하였다. 검증을 위해 주행 제어기의 Stress Test를 진행하였고, Fig. 2의 시간에 따른 제어기 프로세서 온도 데이터를 얻었다. 이를 통해 주행 제어기의 시스템 안정성을 확보하였다.

## 3. 통합 제어 알고리즘

### 3.1. Integration Software Architecture: Hye Ahn

기존 연구의 자율주행 통합 제어 알고리즘의 비체계화로 인해, 분리되지 않은 각 모듈들의 역할, 센서 데이터 메모리 누수 등과 같은 문제점이 발생했다.[3] 본 연구에서는 프로그램의 체계화를 달성하기 위해 Fig. 3과 같은 시스템의 구조를 정립하였다. 크게 각 센서에 동기화되어 통신하고, 센서 데이터를 바탕으로 처리하여 상황을 인지하는 인지(Perception) 단계, 앞의 데이터를 기반으로 주행 경로를 처리 및 결정하는 프로세싱(Processing) 단계, 플랫폼을 제어하여 구동과 조향을 관리하는 제어(Control) 단계로 구분하였다.

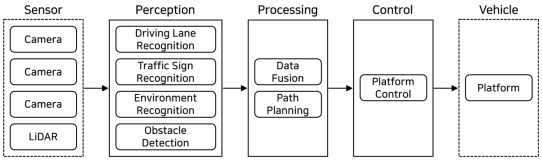



Fig. 3 Structure of the autonomous vehicles system

위의 자율주행 시스템 구조도를 기반으로 통합 프로그램을 제작하였다. 통합 프로그램은 제작할 자율주행 시스템에 필요한 부분들을 모듈화 하여 Fig. 4과 같이 제작하였고, “혜안(Hye Ahn)”이라 명명하였다.

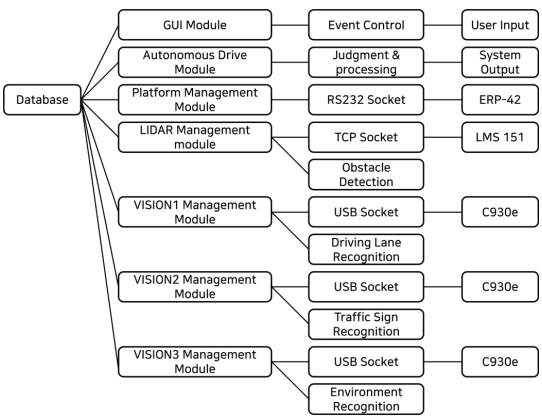



Fig. 4 Software architecture of Hye Ahn program

시스템의 성능적 향상과 체계화를 위하여, 각 센서로부터 데이터를 수집하는 모듈, 수집된 데이터를 처리하는 모듈, 결정된 제어 값을 플랫폼으로 데이터를 전송하는 모듈로 분리하였고, 프로그램은 각 단계를 준수하도록 하였다. 또한, 수많은 데이터를 체계적으로 관리하고자 데이터베이스를 모듈에서 별도로 분리하여 통합 프로그램이 관리하도록 하였다.

### 3.2. Subdivision Module Classification

혜안 프로그램은 7개의 모듈로 구성되어 있으며, 각 모듈은 정해진 역할만을 담당한다. GUI 모듈은 사용자의 자율주행 모드 실행 및 긴급 정지(E-STOP) 등의 입력 처리, 시스템 및 프로그램의 로그와 각 센서와 모듈들의 상태 출력을 담당한다. Autonomous Drive Module은 주행 경로를 결정하고 처리한다. Platform Management Module은 자율주행 플랫폼과의 RS232 통신을 담당하며, 구동 모터의 속도, 기어, 조향각, 브레이크 값, 엔코더 값 등의 제어 값 정보를 송수신 한다. LiDAR Management Module의 경우, LiDAR와의 TCP 통신을 담당하며, 거리 맵 데이터를 송수신 한다. Vision1 Management Module은, 전방 카메라를 통해 데이터를 수신 받아, 차선 추종 알고리즘 수행을 담당한다. Vision2 Management Module은, 전방 카메라를 통해 데이터를 받고, 도로 표지판 인식 알고리즘을 수행한다. Vision3 Management Module의 경우, 측면 카메라를 통해 데이터를 수집하여 보조적으로 주변환경을 인식하는 역할을 담당한다.

## 4. 차선 추종 알고리즘

### 4.1. Improved Lane Tracing: ILT

OpenCV 내장 함수인Canny edge와 Hough transform (C&H)을 결합해 차선 검출을 시도했다.[4] 그러나 1280×720 픽셀 크기의 이미지 처리 속도가 매우 느리고, 곡선 차선 검출이 어렵다는 문제점을 발견하였다. 이를 해결하기 위해 ILT 알고리즘을 고안하였고, 그 처리과정은 아래와 같다.

i. 도로에서의 관심 영역(ROI)을 지정하고 채널 축소(Gray scaling)를한다.

ii. 이미지를 이진화하고 Top view로 변환시킨다.
iii. 특정 좌표축에서 수평으로 이동하며 차선의 기준점을 찾는다.
iv. 기준점에서 상하 두 방향으로 진행하며, 좌우 범위를 조회하여 차선 좌표를 찾아 저장한다.
v. 다음 프레임에서는 이전 기준점 근방에서, 새로운 기준점을 잡아, 위 과정을 반복한다.

그러나, 특정 좌표를 조회하여 차선 여부를 판단할 때, 단일 지점 픽셀 수치만을 이용할 경우, 노면 상의 외란에 취약했다. 이를 해결하고자 조회 좌표 기준, 멀어질수록 낮은 점수를 주는 평가제를 도입하였고, 특정 점수 이상을 차선이라고 판단하도록 하였다.

또한, ILT와 C&H알고리즘의 처리 시간을 비교하였다. 600×400 픽셀 크기의 직선 차선 이미지 처리를 10번 반복하여 걸리는 평균 소요시간을 측정하였다. Table 3에서 알 수 있듯, ILT 알고리즘이 약 2.03배 더 빠르게 수행됨을 확인할 수 있었다.

Table 3 Average processing time of 10 executions

| Algorithm | Processing time(ms) |
|---|---|
| C&H | 87.17 |
| ILT | 43.02 |

### 4.2. Histogram Extension: HIX

ILT 알고리즘을 적용했을 경우, 도로 시험 주행 도중 문제점이 발견되었다. 첫째로, 태양의 밝기나 주변 그림자 상태 등, 환경이 변할 때마다 이진화의 적절한 임계 값을 지속적으로 찾아 수정해주어야 했다. 둘째로, 빛이 많이 반사되는 노면에서는 차선의 일부가 제대로 검출되지 않았다. 이러한 문제를 해결하기 위해 카메라를 통해 들어오는 이미지의 명도를 양극화시키는 HIX 알고리즘을 설계하였다.

i. Top View로 변환한 이미지를 이진화 한 후 흰색 차선 부분을 찾아 변수 'white'에 저장한다.
ii. 도로상의 15개 점의 밝기를 측정하고 평균을 내어 변수 'black'에 저장한다.
iii. 식 (1)을 이용하여 이미지의 모든 픽셀의 명도를 양극화시킨다.

$$value = \frac{P_{i,j} - black + k}{white - black} \times 255 \quad [where\ i = 1, 2, \cdots, ]$$
$$\begin{cases} value \geqq 255 \rightarrow value = 255 \\ value \leqq \ \ 0 \rightarrow value = \ \ 0 \end{cases} \tag{1}$$

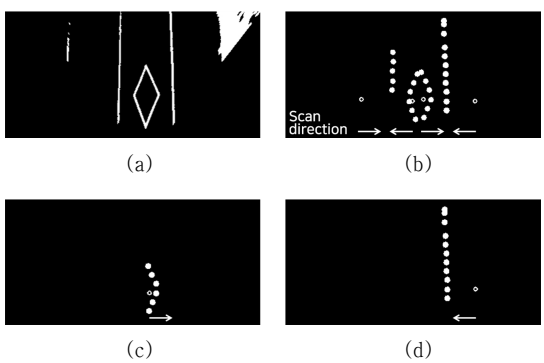
Fig. 5에서 관찰할 수 있듯, HIX 알고리즘을 적용한 결과, 급격한 도로의 밝기 변화에도 차선 인식 정확도가 증가했음을 확인할 수 있었다.

(a) (b)

Fig. 5 (a) Before and (b) after applying HIX algorithm

### 4.3. Enhanced Lane Tracing with Histogram Extension: ELIX

차선 정보를 검출하는 ILT, 환경 변화에 따라 명도를 실시간 보정하는 HIX 두 알고리즘을 통합하여 ELIX를 제작하였다. 주행 테스트 도중 장애물 회피와 같은 돌발 상황 이후, 다시 차선의 정보를 갱신할 때, 차로 내부의 노면 표시나 차선 외부의 표시를 차선으로 인식하여 오판하는 문제가 발생하였다. Fig. 6에서 보여진 바와 같이, ILT에서의 한 방향 검출과 달리, ELIX에서는 네 방향으로 여러 차선 후보를 검출하여 가장 적절하다고 판단되는 차선을 선택하도록 하였다. 위 과정에서는 많은 연산 과정이 요구되지만, 차선을 갱신할 때만 이 연산을 수행하기 때문에 평균 프레임 처리 시간은 거의 증가하지 않으면서 인식 정확도는 크게 향상되었다.




(a) (b)

(c) (d)

Fig. 6 (a) Input Image, (b) ELIX rescan, (c) ILT rescan 1 and (d) ILT rescan 2

## 5. 표지판 인식 알고리즘

### 5.1. LBP Cascade Classifier

표지판을 인식하기 위해 Cascade Classifier를 이용하였으며, 이미지의 모서리, 경계, 평판 영역과 픽셀의 밝기 비교 히스토그램을 사용하는 LBP 방식을 이용해 표지판을 학습시켰다.[4] 실내 및 야외의 다양한 환경, 여러 각도에서 촬영하고, 부분 그림자 및 광원의 다양한 사진 총 500장을 학습에 이용하였다. 표지판 영역의 라벨링(Labeling)을 진행하였고, 각 사진은 30×30 픽셀 크기에서, 30번의 학습을 진행하였다.

실제 환경에서는 반사광에 의한 분류기의 인식을 저해하는 현상이 발생하며, 이렇게 손상된 영역을 복구하기 위해 두 알고리즘을 적용하였다. 먼저, 반사광에 의해 나타나는 이미지의 빛 반사 지점(Glare Point)을 복구하는 NPC 알고리즘을 아래와 같이 제작하였다.

i. 회색조(Gray scale)로 변환된 영상에서 빛 반사 지점을 탐색한다.
ii. 근처 픽셀의 색상 값을 이용하여 빛 반사 지점을 보간 한다.
iii. 빛 반사 지점의 투명도를 추출하고, 화이트 밸런스를 낮춘 후, 인페인팅(Inpainting)을 진행한다.

이미지의 밝기를 일정 수준으로 맞추는 Gradient Histogram (GH) 알고리즘을 아래와 같이 적용하였다.

i. Gray scale로 변환된 영상의 히스토그램을 출력한다.
ii. 출력한 히스토그램의 수치를 평준화하여, 이미지 내의 큰 Edge를 줄인다.

Fig. 7과 같이, 위의 두 알고리즘을 통합하여 이미지의 손상 정도를 낮춤과 동시에, 경계 상자(Bounding Box)를 정확히 위치시킬 수 있게 되었다.

(a) (b)

Fig. 7 (a) Before and (b) after applying NPC algorithm

### 5.2. SVM

표지판 인식을 위해, Support Vector Machine(SVM) Classifier 또한 학습시켜 함께 이용하였다. DSIFT를 이용하여 특징점을 추출하였으며, Fig. 8과 같이 표지판의 종류만큼 결정 트리를 확장하여 모든 표지판에 대해 분류 가능하도록 구현하였다.[5] 학습은 표지판의 다이어그램 부분만을 Fig. 9상의 19×17 grids로 나누어 각 지점에서의 특징점을 추출하였고, 트리의 각 노드를 모두 학습시켜 SVM 결정 트리를 제작하였다.

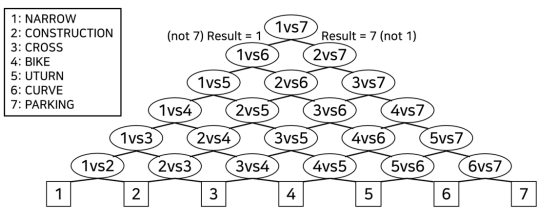



Fig. 8 SVM decision tree

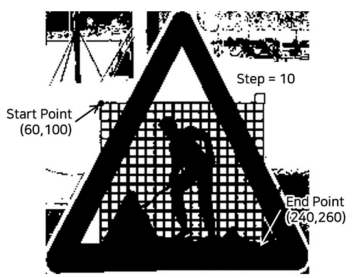



Fig. 9 Grid for SVM feature extraction

### 5.3. YOLO

표지판이 부분적으로 가려지거나 가까운 거리에서 바로 나타나는 경우 등의 상황에서, 더 빠른 인식을 통해 표지판 분류를 확정할 수 있도록, 딥 러닝 기반의 이미지 인식 알고리즘인 Yolo v3를 추가 활용하였다.[6,7] 표지판에 대해 다양한 방면에서 촬영한 각 1500장의 사진 데이터를 라벨링하여, 학습하였다. Batch=64, subdivision=16, width, height=416, learning rate=0.001로 설정하였으며, 40000 steps 학습을 수행하였다. 앞서 사용한 vision 센서를 이용하여 인식을 진행한 결과는 다음과 같다. 표

지판에서 약 4m 떨어진 근거리 지점에서부터 인식을 시작하며, 720p의 해상도에서 최대 60fps의 처리 성능을 보여주었다.

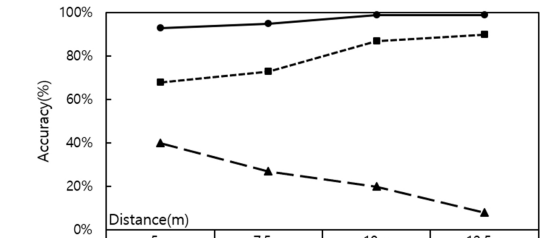
### 5.4. Mutual Validation Procedure: MVP

본 연구에서 제안한 MVP는 2가지 분류기의 값의 비교하여 높은 확률을 나타내는 값을 출력하는 교차 검증 알고리즘이다. 기존과는 차별된 방식으로 데이터 검증 범위에 대한 기준을 상대적으로 설정하는 방식을 도입하였고, 새롭게 고안된 MVP 알고리즘을 통해 성능 측면에서 향상을 이룰 수 있었다.

이를 여러 분류기에 적용하여 테스트를 진행하였고, 추가적으로 성능과 정확도가 높은 객체 인식을 위해, Cascade와 이의 오판을 방지하기 위한 SVM 2가지 방식을 이용한 교차 검증 알고리즘을 제안하였다.

#### 5.4.1. MVP-CS

MVP-CS는 Cascade의 실시간 연산과 SVM의 정확도를 활용한 방식이며, 두 인식기의 값이 연속으로 일관되면 작동된다. 알고리즘이 시작되면 판정 결과와 함께, 인식한 표지판과 플랫폼 간의 거리를 판단한다. 특정 거리가 되면 판정 자료와 함께 거리 데이터를 활용하여 가장 높은 확정 판정을 받은 클래스를 출력한다. Fig. 10에서와 같이, 이 방식을 이용했을 때, 재래적인 인식 결과 비교 방식보다 더 높은 신뢰도를 갖는 것을 확인할 수 있었다. 결과적으로, MVP알고리즘은 객체 판정에 대한 신뢰도 또한 향상시켜 전체적인 인식률이 크게 상승 했음을 알 수 있다.



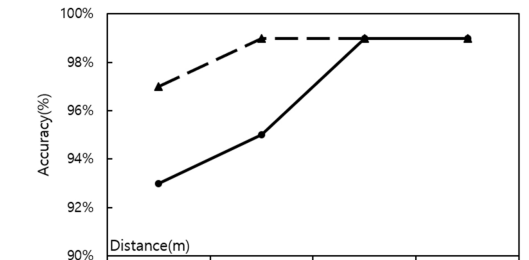

| | 5m | 7.5m | 10m | 12.5m |
|---|---|---|---|---|
| MVP-CS | 93% | 95% | 99% | 99% |
| SVM | 40% | 27% | 20% | 8% |
| Cascade | 68% | 73% | 87% | 90% |

Fig. 10 Accuracy of MVP-CS with conventional method by Distance

추가적으로, CUDA 가속을 통해, 기존 CPU만 활용하였을 때에 비해 1170% 성능 향상을 보였다.[8]

#### 5.4.2. MVP-YOLO

MVP-CS 표지판 인식 알고리즘이 가까운 거리에서 비교적 낮은 인식율을 보이는 이유로 두 객체 인식 알고리즘의 낮은 축적 데이터 양이라고 판단하였다. 이를 보완하기 위해YOLO v3를 도입하여, 근거리에서의 비교적 낮은 인식률을 향상시킨 MVP-YOLO를 추가로 개발하였다.




| | 5m | 7.5m | 10m | 12.5m |
|---|---|---|---|---|
| MVP-CS | 93% | 95% | 99% | 99% |
| MVP-YOLO | 97% | 99% | 99% | 99% |

Fig. 11 Accuracy of MVP-CS and MVP-YOLO by distance

YOLO는 동일 ROI에서 여러 클래스가 출력될 수 있다. MVP-YOLO는 YOLO의 다중 클래스의 결과를 N개의 개별 분류기가 각각의 결과를 출력한 것으로 보고 MVP에 적용하였다. Fig. 11에서 볼 수 있듯이, 그 결과로 근거리에서의 판단 정확도가 MVP-CS에 비해, MVP-YOLO에서 높은 것을 확인 할 수 있다. YOLO는 물체가 부분적으로 가려지거나, 왜곡되었을 경우에도 꾸준한 인식률을 보여주었으나 mean Average Precision(mAP)는 비교적 낮은 수치를 보여준다. 하지만 MVP system을 적용했을 때의 인식 정확도는 99% 정도로 높은 신뢰도를 보여준다.

## 6. 객체 인식 및 회피 알고리즘

### 6.1. Obstacle Detection Algorithm

#### 6.1.1. Data set received from LiDAR

LMS151 LiDAR를 통해, 한 주기에 -45°를 시작으로

270°까지 0.5°간격으로, 총 541개의 거리(R) 데이터를 수신 받는다. 이 극좌표계 데이터를 평면 좌표계로 변환하여 알고리즘에 활용하였다.[9]

6.1.2. Set Region of Interest (ROI) and Object

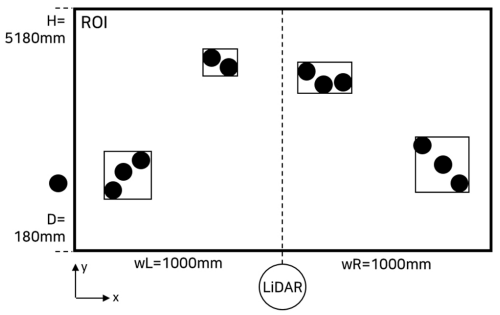



Fig. 12 ROI for obstacle detection

플랫폼 크기를 고려하여, 관심영역을 Fig. 12와 같이 설정하였다. LiDAR를 기준으로 좌측과 우측 너비를 wL, wR로 설정하였고, 본 차량이 지나는 영역을 인지하고 판단할 수 있도록 지정하였다. D는 LiDAR에서 관심영역까지의 거리로, 차체의 범퍼를 외부 물체로 판단하는 경우를 방지하고자 설정하였다. (H-D)는 ROI의 거리로, 이는 Fig. 13에서 계산된 바와 같이, 플랫폼의 최대 조향각 15°를 고려하여 객체를 회피할 수 있는 최소한의 거리를 확보할 수 있도록 지정하였다.

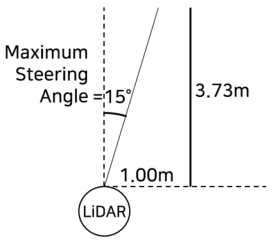



Fig. 13 Setting the length of the ROI

설정한 관심영역 내부의 데이터에 대해서 객체 인식 알고리즘을 적용하였고, 두 점 사이의 거리가 10cm이내이면 하나의 객체로, 이상이면 다른 객체로 판단하여 물체를 인식하였다. 또한 직사각형 형태로 표현된 각 객체는 물체의 크기 및 플랫폼으로부터 물체까지의 상대 위치가 출력되도록 설계하였다.[10]

6.1.3. Offset value

상기 알고리즘을 칼라콘과 같은 뿔 형태의 객체에 적용했을 때, 라이다 센서의 장착된 높이에 따라 인식되는 물체 너비에 차이가 생기는 문제가 발생하게 된다. 이와 같은 문제점을 방지하기 위해 Fig. 14와 같이 실제 객체의 너비와 인식되는 너비 간의 차이를 예측하는 과정을 거쳤고, 너비에 여유 너비(offset)만큼 더하는 방식으로 보완하였다. 이와 같이 다양한 장애물에 대한 여유 너비 값을 저장하여 이용하였다.

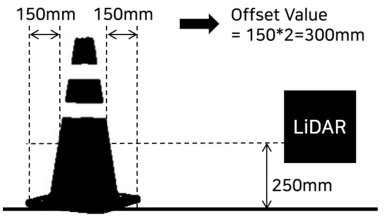



Fig. 14 Setting the offset value

6.2. Obstacle Avoidance Algorithm

6.2.1. Danger Point

위험 요소(danger point)를 판단하고, 위치에 따라 반응하여 객체를 회피하기 위해, Fig. 15와 같이 사각형으로 표현된 각 객체들의 LiDAR에 가장 가까운 점들을 위험 요소의 후보로 설정하였다. 물체가 ROI 중심부에 걸쳐진 경우 객체의 중심 x 값에 따라 위험 요소의 후보 점을 결정하였다. 이로써 다양한 객체에 대한 위험 요소를 설정 및 처리할 수 있었다.

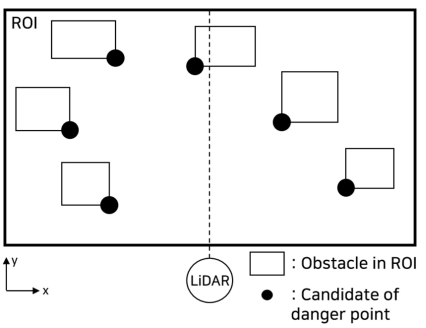



Fig. 15 Setting the candidate of danger point

6.2.2. Steering Angle for Object Avoidance

객체 회피 알고리즘은 Fig. 16처럼, θ만큼 조향 하여 위험 요소를 관심영역밖으로 옮기는 아이디어를 바탕으로 설계되었다. 또한 위험 요소를 포함하는 객체의 중심 x값에 따라 조향 방향이 결정된다.

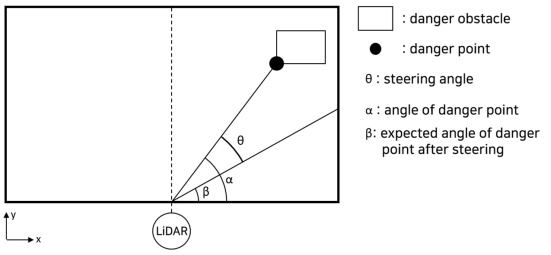



Fig. 16 Operation principle of obstacle avoid algorithm

Fig. 17을 보면, danger point가 A영역(반원 영역) 안에 존재할 경우, β(expected angle of danger point after steering)를 판단할 수 없다는 것을 알 수 있다. 즉, 위의 알고리즘이 적용되지 않는 경우에 해당할 수도 있기에, 이 경우에는 최대 조향각에서 회피알고리즘을 적용하였다. 또한, 그 중에서도 B영역에 위험 요소가 존재하여, 차체 정면에 객체가 인지될 경우에는, 차의 움직임을 멈추는 방식으로 알고리즘을 보완하였다.

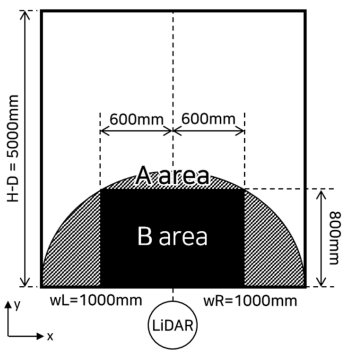



Fig. 17 The areas where avoidance algorithm cannot work

## 7. 결 론

본 논문에서는 먼저 비전 센서를 통한 향상된 차선 추종 알고리즘을 개발하였다. 차선 인식 알고리즘인ILT와 주변 환경 밝기 조절 알고리즘인 HIX 를 결합 및 보완하는 과정을 거쳐 최종적으로 ELIX 알고리즘이 완성되었다. 기존Hough transform알고리즘 대비, 유연한 환경적응성과 가벼운 연산량을 지닌 알고리즘임을 확인할 수 있었다. 다음으로, 비전 센서를 통한 표지판 인식 알고리즘을 새롭게 제안하였다. 기존에 개별적으로 사용되었던 LBP Cascade Classifier, SVM을 결합 및 교차 검증하는 MVP-CS알고리즘을 새롭게 고안하였다. MVP내에서 정확도 또는 속도가 상대적으로 부족한 기존의 두 가지 알고리즘을 병행하고 상호 교차검증과정을 거침으로써 인식률이 향상되었다. 더 나아가, SVM 등에 비해 인식률과 속도가 더 높은 알고리즘을 결합 대상으로 삼음으로써 획기적인 성능 향상을 기대할 수 있게 되었다. 실제로 YOLO를 새로 도입한 MVP-YOLO의 경우 MVP-CS와 비교하여 정확도가 5m기준 4% 향상됨을 보일 뿐만 아니라 항상 MVP-CS를 뛰어넘었다. 이를 통해, 새로운 인식 알고리즘을 개발하는 것 외에도, MVP를 통한 기존 알고리즘들 간의 적절한 융합 및 교차 검증으로 더 나은 성능 향상을 이끌어냄을 보여주었다. 이와 함께, Lidar센서를 통한 물체 감지 알고리즘으로 장애물을 인식한 후, 장애물의 위치를 모델링하여 물체 회피 알고리즘을 통해 최적의 회피 각도를 출력하였다.

그러나, 본 논문에서 제안한 비전을 이용한 차선 추종 알고리즘은 차선이 급격하게 바뀌거나 교차로 등에서는 성능이 저하되므로, 이에 대한 추가 연구를 할 계획이다. 또한 물체 인식 알고리즘을 위해LiDAR 센서 단독으로 사용했기 때문에 물체를 피하기 위한 여유 너비를 크게 설정해 놓았지만, 추후 연구에서는 비전 센서와 융합하여 물체에 따라 다른 안전 거리를 최적화하는 제어 알고리즘을 개발할 계획이다.

## 참고문헌

(1) 정동규, 김선형, 2016, “지능형 자동차 현황과 향후 전망”, 한국정보기술학회지, Vol. 14, No. 2, pp. 21~26.

(2) 이민채, 한재현, 장철훈, 선우명호, 2013, “영상 및 레이저레이더 정보융합을 통한 자율주행자동차의 주행환경인식 및 추적방법”, 한국지능시스템학회지, Vol. 23, No. 1, pp. 35~45.

(3) Choi, G., Lee, C., Lee, J., Lim, Y., Seo, J. and Shin, k., 2018, “Study on Sensors and Algorithms for Autonomous Vehicle”, ESIT, Vol. 3, No. 80.

(4) Chang, F., Chen, Z, 2014, and Liu, C., “Rapid

Multiclass Traffic Sign Detection in High-Resolution Images", IEEE Transactions on Intelligent Transportation Systems, Vol. 15, No. 6, pp. 2394~2403.

(5) Ahn, H. and Kim, J., 2011, Corporate credit rating using multiclass classification models with order information. World Academy of Science, Engineering and Technology, Vol. 5, No. 12, pp. 1783~1788.

(6) Zhu, Z., Liang, D., Zhang, S., Huang, X., Li, B., Hu, S., 2016, "Traffic-Sign Detection and Classification in the Wild", IEEE Conference on Computer Vision and Pattern Recognition, pp. 2110~2118, doi:10.1109/cvpr.2016.232.

(7) Farhadi, A. and Redmon, J., 2018, "YOLOv3: An incremental improvement", arXiv: 1803.10827.

(8) Timothy Masters, 2015, "Deep Belief Nets in C++ and CUDA C", Apress, Vol. 3.

(9) 이호준, 채흥석, 서호태, 이경수, 2018, "자율주행을 위한 레이더 기반 인지 알고리즘의 정량적 분석", 한국자동차안전학회, Vol. 10, No. 2, pp. 29~35.

(10) Han, J., Kim, D., Lee, M. and Sunwoo, M., 2012, "Enhanced road boundary and obstacle detection using a downward-looking LiDAR sensor", IEEE Transactions on Vehicular Technology, Vol. 61, No. 3, pp. 971~985.