YOLOPv2 를 활용한 차선 탐지 기반 자율주행 구현

박준혁 [1], 이재인 [1], 정예찬 [1], 이시우 [2], 전재욱 [3]
[1] 성균관대학교 전자전기공학부 학부생
[2] 성균관대학교 소프트웨어학과 석사과정
[3] 성균관대학교 전자전기컴퓨터공학과 교수

rchkl2380@gmail.com, jglee6517@g.skku.edu, ankylo7281@g.skku.edu, edenlee@g.skku.edu, jwjeon@skku.edu

Autonomous Driving Implementation Based on Lane Detection Using YOLOPv2

Joon-Hyuk Park[1], Jae-In Lee[1], Ye-Chan Jung[1], Si-Woo Lee[2], Jae-Wook Jeon[3]
[1]School of Electronic and Electrical Engineering, Sung-Kyun-Kwan University
[2]Dept. of Computer Science and Engineering, Sung-Kyun-Kwan University
[3]Dept. of Electrical and Computer Engineering, Sung-Kyun-Kwan University

요 약

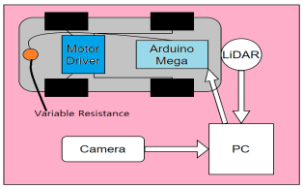
본 연구에서는 전동 차량의 자율주행 기능 구현을 위한 방법을 제시한다. 주요 구성 요소로는 카메라, PC, 아두이노, 모터드라이브, 가변저항 등이 사용되었다. 카메라를 통해 데이터를 수집한다. YOLOPv2 lane detection 딥러닝 모델을 사용하여 차선을 탐지하고, 후처리 과정을 통해 주행 경로를 정확히 인식한다. RANSAC 알고리즘을 활용하여 outlier 에 강건한 2 차 함수 회귀를 수행하고, 이를 바탕으로 주행 중 필요한 정보를 파악한다. 이러한 정보를 바탕으로 차량의 조향각을 조절하여 안전하고 효율적인 자율 주행을 구현하였다.

1. 서론

자율주행 차량은 현대 교통 시스템에서 가장 혁신적인 발전 중 하나로 간주되며, 그 성능과 안전성은 연구와 개발에 따라 지속적으로 향상되고 있다. 기존의 연구에서는 다양한 센서와 알고리즘을 사용하여 차량의 주행을 자동화하려 했지만, 본 연구에서는 전동 차량의 자율주행과 주차 기능 구현을 위해 특정 하드웨어 및 알고리즘 조합을 탐구한다. 본 논문에서는 카메라를 활용한 주변 환경 인식, 그리고 YOLOPv2 lane detection 딥러닝 모델을 이용한 주행 경로 검출과 조향 알고리즘에 대해 설명한다.

2. 시스템 구성

전동 차량의 자율주행 기능 구현을 위해 카메라, PC, 아두이노, 모터드라이브, 가변저항을 사용하였다. (그림 1)에서는 전반적인 시스템의 구성을 보여준다. 카메라를 통해서 주변의 데이터를 얻고, PC 의 Python 프로그램을 통해 처리하여 제어 정보를 아두이노에 시리얼 통신으로 넘겨준다. 아두이노에서는 넘어온 제어 정보를 바탕으로 가변저항을 통해 조향각을 인지 후 제어하고, 모터의 속도도 제어하게 된다. 카메라는 APC920W 를 사용했다. (표 1)에서 카메라의 화소수, 해상도, 프레임, 시야각을 확인할 수 있다.



(그림 1) 차량 제어 구성도

<표 1> APC920W 주요 제원

화소수 (H/W)	400 만	비디오 해상도	2560x1440 (QHD)
비디오 프레임	30fps	시야각	110°

3. 자율주행 구현 방법

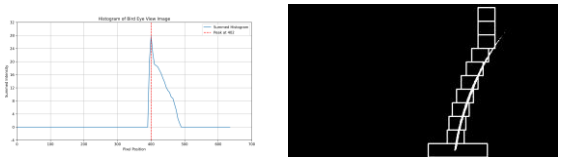
Lane 에 대한 정확한 정보를 얻기 위해서 lane detection 용 딥러닝 모델을 사용했다. BDD100K 데이터셋에서 높은 정확도를 보이는 YOLOP 모델의 후속 버전인 YOLOPv2 를 선정하였다 [1]. (그림 2)에서는 딥러닝 모델을 통해 얻어진 lane 에 대한 이진화 마스크를 보여준다.

(그림 2) YOLOPv2 을 통한 lane 마스크.

Lane 마스크에서 필요한 정보만 얻기 위한 후처리 과정이 필요하다. (그림 3)에서는 전반적인 후처리 과정을 보여준다. 마스크 이미지를 후처리 과정을 거쳐 수평적인 성분을 제거하고 두꺼운 lane 을 실제의 크기와 비슷하게 만들어준다. Lane 이미지를 위에서 아래로 내려다보는 형태인 bird eye view 형태로 변환한다. 이후 ROI(region of interest)를 설정하여 실선만 남긴다.

(그림 3) lane 마스크 후처리 과정.

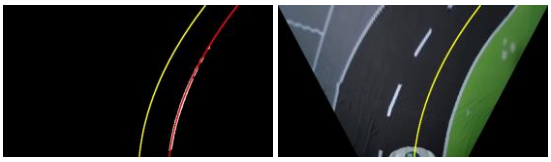
(그림 4)에서는 추가적인 잡음을 제거하기 위한 방법을 보여준다. histogram 을 통해서 sliding window 의 시작 위치를 지정한다. sliding window 방법을 통하여 원하는 lane 에 대한 픽셀만 추출한다.



(그림 4) histogram 방식과 sliding window 방식.

Lane 의 픽셀을 바탕으로 2 차 함수 회귀를 통해, 2 차 함수의 계수를 구한다. (그림 5)의 빨간색 선은 실제 Lane 에 대한 픽셀들로 2 차 함수를 회귀한 것이고, 노란색 선은 빨간색 선을 바탕으로 주행하는 Following line 을 그린 것이다. 이때 2 차 함수는 $x = ay^2 + by + c$ 꼴로 표현된다. x축은 가로축, y 축은 세로축을 의미하며 (0,0) 좌표는 사진의 좌상단이다.

이때 사용되는 2 차 함수 회귀 알고리즘으로 RANSAC 알고리즘을 사용하였다. RANSAC 은 특정 임계값 이상의 데이터를 완전히 무시해버리는 특성이 있어 outlier 에 강건한 특징을 보여주기 때문에 도로의 lane detection 을 위해 사용된 연구도 있다 [2].



(그림 5) Following line 생성.

로직에 따라 얻어진 2 차 함수를 바탕으로 주행에서 필요한 정보(Following line 과의 차의 중심과의 차이, 각도, 곡률)를 얻는 과정을 설명하면 다음과 같다.

1. 정보를 얻기 위한 2 차 함수의 y_1 값을 정한다.
2. y_1 을 2 차방정식 $x = ay^2 + by + c$ 에 대입하여 Following line 의 위치 x_1을 얻고 차의 위치인 화면의 중심 과의 차이를 계산하고 차이에 의한 조향각을 구한다.
3. y_1에서의 각도를 알아내기 위하여 미분을 진행한다. $angle = 2ay + b$ 계산 과정을 통하여 각도를 알아내고 각도에 의한 조향각을 구한다.
4. 2 차 함수의 곡률 구하는 식 (1)을 활용한다. 해당 식에 대입하여 곡률을 얻어낸다.
5. 곡률의 역수를 취해 회전반경을 계산하고, 픽셀단위의 회전반경을 미터단위로 변환해준다. 회전반경과 차체의 축거를 바탕으로 곡률에 의한 조향각을 계산해 낸다.

6. 각각의 조향각에 적절한 가중치를 곱해서 최종적인 조향각을 도출한다.

$$k(y) = \frac{f''(y)}{((f'(y))^2+1)^{\frac{3}{2}}} \qquad (1)$$

4. 결론

(그림 6)과 같이 전동 차량 제어 시스템을 구현하고, 딥러닝 모델과 이미지 처리 알고리즘을 사용하여 전동 차량에서의 자율주행을 구현하였다. 현재는 실제 환경과 차이가 있지만, 차후 이를 발전시킨다면 실제 차량에도 적용할 수 있을 것이다.

(그림 6) 전동 차량에서의 자율 주행 구현.

ACKNOWLEDGMENT

이 논문은 정부(교육부-산업통상자원부)의 재원으로 한국산업기술진흥원의 지원을 받아 수행된 연구임 (P0022098, 2023 년 미래형자동차 기술융합 혁신인재양성사업)

참고문헌

[1] Wu, Dong, et al. "Yolop: You only look once for panoptic driving perception." *Machine Intelligence Research*, 19. 6, 550-562, 2022

[2] J. Guo, Z. Wei and D. Miao, "Lane Detection Method Based on Improved RANSAC Algorithm," 2015 IEEE Twelfth International Symposium on Autonomous Decentralized Systems, Taichung, Taiwan, 2015, pp. 285-288, doi: 10.1109/ISADS.2015.24.