PID 제어를 이용한 자율주행자동차의 차선 추적

김현식, 장재영, 김찬수, 전중남
충북대학교 소프트웨어학과
email: 03tlqfk@naver.com, lunid0132@naver.com, wsfeed33@naver.com, joongam@cbnu.ac.kr

Lane Tracking of Autonomous Vehicles using PID Control

Hyun-Sik Kim, Jae-Young Jang, Chan-Su Kim, Joongnam Jeon
Dept. of Computer Science, Chungbuk National University

요 약

자율주행 자동차는 운전자가 개입하지 않고 차량에 부착된 다양한 센서를 통해 얻은 데이터를 기반으로 차량을 스스로 제어하며 설정한 목적지까지 주행한다. 본 논문에서는 단일 카메라와 영상을 사용한 차선 검출하고 추적하는 방법을 제안한다. 영상의 하단 부분만 분리하여 차선을 검출하기 위하여 외곽선 검출 과정을 거친 후 허프 변환을 통해 양 차선의 중심을 구한다. 이 값을 바탕으로 PID 제어로 차량의 차선을 유지한다. 모형 차량과 모형 트랙에서 차선 인식 후 차선을 추적하여 주행하는 동작을 시험하였다. PID 제어를 위한 적정 각 항의 값을 구하였다. 시험 결과 차선 검출 알고리즘은 성공적으로 동작함을 확인할 수 있었다.

1. 서론

자율주행 자동차란 운전자가 차량을 운전하지 않아도 주변 환경을 인식하고, 주행 상황을 판단하여, 차량을 제어함으로써 스스로 주어진 목적지까지 주행한다.

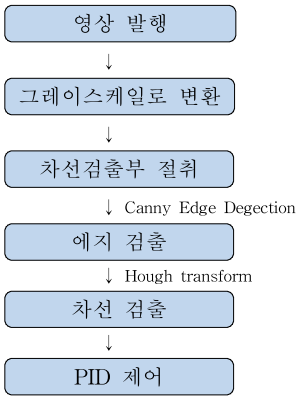
최근에는 이러한 자율주행 자동차가 교통사고를 줄이고, 교통 효율성을 높이며, 연료를 절감하고, 운전을 대신해줌으로써 편의를 증대시킬 수 있는 미래 개인 교통 수단으로 주목 받고 있다[1]. 자율주행을 위한 차선 인식과 차선 유지 기술 연구 또한 지속적으로 연구되고 있다. 기존의 차선 인식 방법 중 하나는 도로의 차선에 자기 센서를 심는 방식[2]이다. 이 방식은 주변의 자기 물질로 인한 오작동의 발생이 높고 센서 매설 및 유지 비용이 막대하다는 단점이 있다. 딥러닝을 이용한 차선 검출 방법도 연구[3]되어 있다. 정확도가 높지만 딥러닝 특성상 연산량이 비교적 많고, 학습을 시켜야하는 단점이 있다. 영상처리를 이용한 방식[4]은 많이 사용되고 있다.

본 논문에서는 영상처리를 이용한 차선 검출 기법을 연구하였다. 단일 카메라의 영상을 취득하여 라즈베리파이 상에서 허프 변환과 PID(Proportional-Integral-Differential) 제어를 통해 차량의 차선 인식 및 이탈을 방지하는 기법을 제안한다.

2. 차선제어 시스템 소개

2.1 제어 시스템 구조

주행 중인 차량에서 취득된 영상에서 OpenCV[5]의 영상처리 기법을 통해 차선의 좌측과 우측 선분을 구한다. 이를 통해 차선의 중심값을 구할 수 있다. 구한 값에 PID제어 기법을 사용하여 차량의 조향을 제어한다. ROS[6] 데이터의 흐름은 그림 1과 같다.

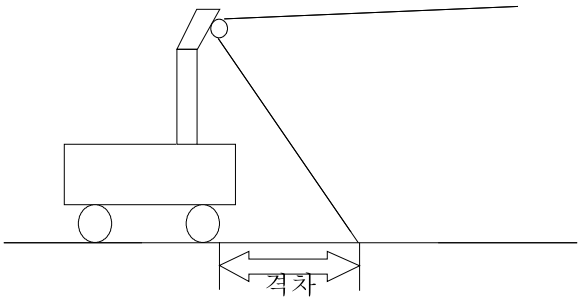


(그림 1) 제어 시스템의 흐름

2.2 자동차의 시점

그림 2와 같이 실제 자동차의 위치와 카메라 화면에 보이는 영상의 위치에 격차가 존재한다. 본 논문에서 사용하는 카메라의 수직각은 40°이며 카메라가 차체의 영상을 잡지 않기 위해, 높이를 올리면, 격차가 더 길어져 지연시간이 오래 걸린다. 이 지연시간을 줄이기 위해 카메라의 시선을 아래쪽을 향하면 격차가 줄어들게 상기 문제를 해

결하였다. 하지만 차체가 검출되면 차선의 영상을 잡는 것이 판독에 오류가 생길 수 있어, 차체를 검출하지 않기 위해 카메라의 각을 약 30도로 꺾었다. 그렇게 발생한 격차는 약 23.25cm이다. 컴파일을 위해 약 0.23초의 시간이 소모되며, 본 논문에서 사용한 차량의 속도는 약 11.45cm/s이므로, 약 2.63cm를 이동한 후 움직임이 결정된다. 따라서 약 20.62cm의 거리의 격차가 발생한다. 이 격차를 해결하기 위해 지연시간을 주어 해당 격차는 해결할 수 있다.



(그림 2) 자동차의 시점

2.3 차선 제어 알고리즘

카메라에서 발행한 영상을 구독하여 그레이스케일로 변환 후 차선을 검출할 수 있는 영상 하단 부분을 자른다.

잡음 제거에 우수한 캐니 에지 알고리즘을 이용하여, 에지를 검출한다. 자른 이미지는 캐니 에지 알고리즘으로 외곽선을 검출한다. 캐니 에지 알고리즘은 표 1과 같은 단계를 거친다.[7]

<표 1> Canny Edge Detection의 단계

Step 1	Noise Reduction
	이미지에 노이즈가 있으면 에지를 찾는데 어려울 수 있어, 가우시안필터를 이용해 이미지의 노이즈를 감소
Step 2	Edge Gradient Detection
	Sobel 커널을 수평방향, 수직방향으로 적용하여 각 방향의 기울기를 획득
Step 3	Non-maximum Suppression
	에지에 기여하지 않은 픽셀을 제거하기 위해 이미지 전체를 스캔
Step 4	Hysteresis Thresholding
	Step 3를 거친 것들이 실제 에지인지 아닌지 판단

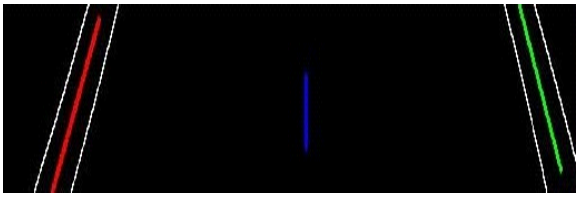
이후 확률적 허프 변환으로 직선을 구한다. 이 방법으로 2차원 상에 있는 외곽선의 두 점을 구할 수 있다. 허프 변환은 이미지에서 모양을 찾는 방법으로 본 논문에서는 차선의 모양을 찾는데 사용하였다 기본적으로 허프 변환은 직선의 방정식을 이용하는데, 극 좌표계를 이용한 삼각함수로 표현하며 식(1)과 같다.[8]

$$r = xcos\theta + ysin\theta \qquad (1)$$

두 점은 4개의 정수형 변수로 이루어져 있는 배열 형태로 나타나는데, 영상 처리 시 좌측에서 우측으로 연산하기 때문에 배열의 첫 번째, 두 번째 원소는 좌측 점의 x, y 좌표이고, 세 번째, 네 번째 원소는 우측 점의 x, y 좌표이다. 이러한 점을 이용하여 첫 번째 원소가 영상의 x축 중심값보다 작으면 좌측 차선 선분, 세 번째 원소가 영상 x축 중심값보다 크면 우측 차선 선분으로 분류한다. 좌측 차선으로 분류된 선분들의 왼쪽 점 x축 값의 평균을 구한다. 우측 차선 선분들은 오른쪽 점의 x축 값의 평균을 구한다. 이 평균값들을 합하여 2로 나누면 차선의 중심값을 구할 수 있다. 이 값은 ROS를 통하여 발행한다. 그림 3은 상기내용의 순서를 가시적으로 표현한 것이다.

Step 1. 카메라 영상

Step 2. 차선 검출부



Step 3. 허프 변환

(그림 3) 영상 처리 순서

차선의 중심값으로만 주행을 할 시 차량이 매우 요동치기 때문에 필터링과 PID 제어를 통해 차량을 제어한다. 차선의 중심값의 오차를 줄이기 위해 일정 크기만큼 다른 배열에 저장 후 최댓값과 최솟값을 제외한 평균값을 계산한다. 차량의 카메라 영상과 실제 움직이는 차선에는 작은 거리차가 있어 평균값을 배열에 넣어 지연 처리하게 한다. 지연된 값을 PID 제어하여 차량의 조향을 결정한다.

PID 제어식은 식(2)와 같다.[9]

$$MV(t) = K_p e(t) + K_i \int_0^t e(t)\,dt + K_d \frac{de}{dt} \qquad (2)$$

K_p = 비례항, K_i = 적분항, K_d = 미분항, $e(t)$ = '차선의 중심값 - 영상의 중심값'으로 설정한다.

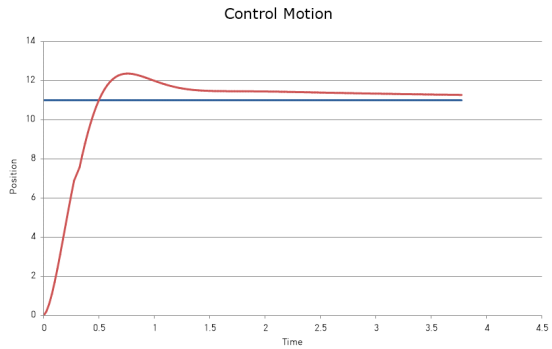
비례항과 적분항이 너무 높으면 주어진 값에 큰 변화를 초래하며 시스템이 불안정할 수 있다. 반대로, 비례항이 너무 작으면 응답성이 떨어지거나 민감하지 않은 제어기가 되며, 제어 조치가 너무 작을 수 있다.

미분항이 너무 높으면 민감하지 않은 제어가 되며, 제어 조치가 너무 작을 수 있다. 반대로 너무 작으면 초기 시스템이 불안정 하며, 커브길 진입 시 다시 시스템이 불안정

하다. 따라서 적절한 값을 선정하는 것과, 적절한 균형을 이루어야 한다. 설정된 값을 통해 차량의 조향을 결정한다. 이 값을 ROS를 이용해 서보모터로 발행한다.

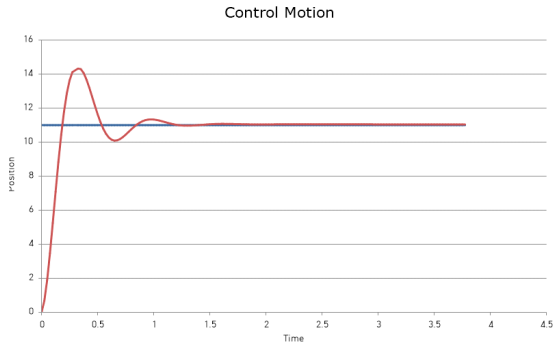
3. 시험 구동

PID 제어 기법을 시험하기 위해 카네기 맬런 대학교에서 만든 PID시뮬레이터[10]의 기본 값인 K_p = 3, K_i = 0.8, K_d = 0.7을 대입 하였을 때는 그림 4와 같이 나타났다. Position은 차량의 위치를 나타내는 것이고, Time의 단위는 sec(초)를 나타낸다. 실제 검증을 위해 만든 트랙의 차폭을 22cm 로 만들었기에, 오차값은 11cm로 지정 한 결과이다. 하지만, 제어값이 목표치에 근사하지만 시간이 지나도 제대로 가까워지지 않기 때문에 해당 K_p, K_i, K_d 값은 적절하지 않다고 할 수 있다.



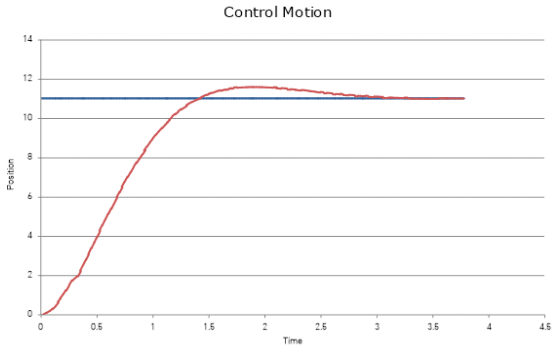
(그림 4) K_p = 3, K_i = 0.8, K_d = 0.7 일 때 차량의 위치변화 그래프

K_p의 값을 늘리면 추후 안정성이 좋기에 K_p = 10, K_i = 0.8, K_d = 0.7을 대입한 결과는 그림 5와 같다. 약 1.2초 이후 목표치에 근사하는 것을 볼 수 있으나, 초기에 제어값이 목표치와 너무 멀어지기 때문에, 적정하지 않다고 할 수 있다.



(그림 5) K_p = 10, K_i = 0.8, K_d = 0.7 일 때 차량의 위치변화 그래프

그림 6와 같이 실험을 통해, P값의 과도한 불안정성이 구동 초기 차선이탈의 원인이 된다. 그래서 비례항의 값을 오히려 점차적으로 감소시키며 진행하였다. K_p = 0.5, K_i = 0.0003, K_d = 0.3일 때 제어값의 형태가 위험이 적을 것으로 보인다. 적분항의 값을 미미하게 부여한 이유는 적분항의 값이 오히려 잡음이 섞여 파형이 생성되는 모습을 보인다. 결과는 그림 6과 같다. 그래프의 형태가 목표치에 근접하며, 제어값이 목표치에서 멀어지지 않는 것을 확인할 수 있다. 따라서 이 K_p, K_i , K_d 값들을 사용하여, 차선 추적이 가능하다.

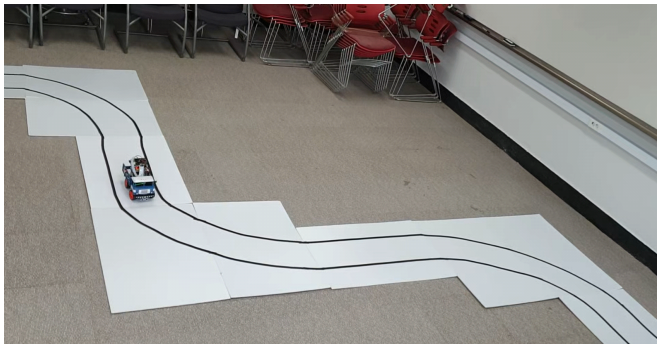


(그림 6) K_p = 0.5, K_i = 0.0003, K_d = 0.3 일 때 차량의 위치변화 그래프

그림 6에서 도출된 K_p = 0.5, K_i = 0.0003, K_d = 0.3을 이용하여, 실제 시험을 해보았고, 결과는 전부 차선을 이탈하지 않고 완주에 성공하였다. 그림 7은 이번 시험에 사용한 모형 차량이다. ROS[6]를 탑재한 라즈베리파이를 제어기로 사용하고, 2개의 DC모터와 한 개의 서보모터를 통해 바퀴를 구동한다.

(그림 7) 시험 주행용 모형 차량

그림 8은 시험에 사용한 트랙이다. 트랙은 폭이 약 22cm 이고, 길이는 약 5.5m, 차선의 두께는 1.9mm이다.



(그림 8) 시험 트랙 및 사진

4. 결론 및 향후연구

본 논문에서는 PID 제어와 영상처리를 활용하여 단일 카메라를 사용한 차선 검출 및 추적 방식을 제안하였다. 허프 변환을 통한 잡음제거 및 에지 검출을 진행하고, 검출된 에지를 이용하여 PID 제어 기법을 사용하였다. PID 제어 기법을 사용하기 위해, 경험적 학습기법을 통해 제어값을 도출 하였다.

도출된 제어값의 검증을 위해 제작한 트랙에서 주행을 수행하였다. 모형 차량이 직선 도로와 연속된 커브길을 주행하도록 하였다. 해당 시험에서 차선 검출 알고리즘이 성공적으로 동작함을 확인할 수 있었다.

향후 연구에서는 제어기법을 더 발전시킬 경우 실제 차량에 적용 가능성이 있어보인다.

Acknowledgement

이 연구는 과학기술정보통신부 및 정보통신기획평가원의 SW중심대학 사업의 연구결과로 수행되었음" (2019-0-01183)

참고문헌

[1] 안경환, 이상우 , 한우용 , 손주찬. 자율주행 자동차 기술 동향. 전자통신동향분석, 28(4): 35-44, 2013.

[2] 김명준, 김의선, 류영재, 임영철 자기 센서 방식 자율주행 차량의 경로 인식 성능 개선. 센서학회지, 12(5), 211-217. 2003.

[3] 이동현, 이경민, 최병호, 인치호. 딥러닝을 이용한 적응적 차선 검출 알고리즘. 대한전자공학회 학술대회, 1571-1572. 2018

[4] 문종술, 박정인, 백석영, 성우석. Robot Operating System 을 활용한 자율주행 모형차 기반의 차선 인식 알고리즘 연구. 제어로봇시스템학회 국내학술대회 논문집, 137-139. 2018

[5] OpenCV, https://opencv.org/, 12-09-2019

[6] ROS.org|rowering the world, http://www.ros.org/, 14-09-2019

[7] J F Canny, "A computational approach to edge detection." IEEE Transactions on Pattern Analysis and Machine Intelligence, 8(6):679-698, 1986.

[8] https://en.wikipedia.org/wiki/Hough_transform/, 21-09-2019

[9] https://en.wikipedia.org/wiki/PID_controller/, 21-09-2019

[10] 카네기 멜런 대학교 - PID 시뮬레이션, https://www.cs.cmu.edu/afs/cs/academic/class/15883-f15/lectures/cerebellum-controller/, 16-09-2019