라즈베리파이와 OpenCV를 활용한 선형 검출 알고리즘 구현

이성진[1,*] · 최준형[2] · 최병윤[2]

[1]부산외국어대학교 · [2]동의대학교

Implementation of Linear Detection Algorithm using Raspberry Pi and OpenCV

Sung-jin Lee[1,*] · Jun-hyeong Choi[2] · Byeong-yoon Choi[2]

[1]Busan University of Foreign Studies · [2]Dongeui University

E-mail : sjlee@bufs.ac.kr / choijunhyeong@naver.com / bychoi@deu.ac.kr

요 약

자율주행 연구가 활발히 진행되면서 ADAS(Advanced Driver Assistance System)에서 차량의 위치를 파악하고 경로를 유지하는데 차선 검출은 필수적인 기술이다. 차선 검출은 허프 변환과 RANSAC(Random Sample Consensus)과 같은 영상처리 알고리즘을 이용하여 검출한다.

본 논문은 라즈베리파이3 B+에 OpenCV를 이용하여 선형 도형 검출 알고리즘을 구현하고 있다. OpenCV 가우시안 블러 구조와 캐니 에지 검출을 통해 문턱값을 설정하였고, 선형 검출 알고리즘을 통한 차선 인식에 성공하였다.

ABSTRACT

As autonomous driving research is actively progressing, lane detection is an essential technology in ADAS (Advanced Driver Assistance System) to locate a vehicle and maintain a route. Lane detection is detected using an image processing algorithm such as Hough transform and RANSAC (Random Sample Consensus).

This paper implements a linear shape detection algorithm using OpenCV on Raspberry Pi 3 B+. Thresholds were set through OpenCV Gaussian blur structure and Canny edge detection, and lane recognition was successful through linear detection algorithm.

키워드

ADAS, RANSAC, Raspberry Pi, Linear detection

Ⅰ. 서 론

자율주행 차량은 4차 산업 혁명과 자동차 기술 발전과 함께 사회적 관심이 증가하고 있는 추세이다. 자동차의 성능이 향상되는 과정에서 자율주행 기능 역시 발전이 진행되고 있으며, 이에 따른 시장 규모의 증가로 인해 4차 산업혁명에서 중요한 역할을 차지할 것으로 예상되고 있고, 특히 자율주행 자동차의 중요한 기술인 교통사고 방지나 효율적인 교통흐름 등이 구현되어, 이에 대한 시장의 변화가 급격한 발전을 할 것으로 예상된다[1].

자율주행 차량은 자율 주행의 정도를 적용하여 단계별로 진행되고 있으며[2], 구글을 비롯하여 벤츠, 포드, 토요타, 현대기아자동차 등에서 자율 주행에 필요한 부분적 기술 개발을 완료 및 진행 중에 있다. 자율주행 관련 연구는 국가 단위로 기업, 대학이 공동으로 참여하여 진행되고 있으며, 자율주행에 대한 기술개발뿐만 아니라 실제 주행실험을 통해 교통흐름과 에너지 효율을 높이기 위한 목적의 연구들을 수행하고 있고, 자동차 간의 효율

* speaker and corresponding author

적인 군집 주행과 자율주행 자동차 전용 도로에 활용될 수 있는 센서 관련 기술, 도로의 구성요소 및 기반시설 등 자율주행 자동차의 구현을 위한 기술 중심의 연구가 활발히 진행되고 있다[3-4].

II. OpenCV

OpenCV(Open Source Computer Vision)는 실시간 컴퓨터 비전을 처리하기 위해 만들어진 크로스 플랫폼 라이브러리로 윈도우, 리눅스 등에서 오픈소스로 사용할 수 있다. OpenCV는 기본적으로 C++로 구현되어 있으나, 파이썬, 자바, C#, 루비 등 다른 언어에서도 사용할 수 있고, 이미지, 영상처리 Object Detection, Motion Detection 등의 기능을 지원하고 있다.

OpenCV에서 이미지나 영상을 처리하기 위해서는 가우시안 블러, 케니 알고리즘, 마스크 처리, 영역 추출, HoughLine 등을 사용한다[5]. 가우시안 블러는 가우시안 함수를 이미지에 합성곱을 해주는 개념으로, 이미지를 필터링화 하는데 사용한다. OpenCV에서는 kernel을 적용하여 블러 처리를 진행하는데, 기본적으로 5×5 행렬로 필터를 적용하고 있다. 가우시안 블러링은 픽셀값에 동일한 가중치를 부여하는 평균 블러링과 다르게 <그림 1>과 같이 중심에 있는 픽셀에 좀 더 높은 가중치를 부여한다.

기본적으로 블러링은 에지를 포함하여 전체적으로 블러링 되지만, 에지가 남아있는 상태에서 블러링이 진행되므로, 캐니(Canny)로 에지를 검출하기 전 노이즈를 제거할 때 사용된다.

$\frac{1}{273}$

1	4	7	4	1
4	16	26	16	4
7	26	41	26	7
4	16	26	16	4
1	4	7	4	1

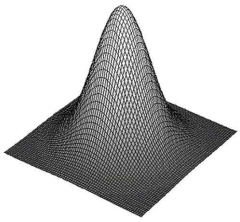

그림 1. OpenCV 가우시안 블러 구조[6]

III. 캐니 에지 검출

캐니 에지 검출은 1986년 John F Canny가 개발한 알고리즘으로, 노이즈 제거, 높은 그라디언트 값 추출, 에지에 영향이 없는 픽셀 제거, 문턱 값 판단의 4단계로 구성되어 있다.

노이즈 제거는 <그림 1>에서 언급한 가우시안 필터를 이용하여 이미지의 노이즈를 줄여준다.

가우시안 필터로 노이즈가 제거된 이미지에서 에지 검출에 사용하기 위해 기울기 값을 추출할 필요가 있다. 수평방향의 기울기를 Gx, 수직방향의 기울기를 Gy로 지정할 경우, 픽셀 (x, y)에서 검출할 수 있는 기울기의 값은 <수식 1>을 통해 획득할 수 있다.

$$Edge\,Gradient(G) = \sqrt{G_x^2 + G_y^2}$$
$$\angle(\theta) = \tan^{-1}\left(\frac{G_y}{G_x}\right)$$

수식 1. 기울기 검출 계산식

에지에 기여하지 않은 픽셀을 제거하기 위해 이미지를 모두 스캔하고, 그라디언트 최대값을 가진 픽셀을 도출한다. <그림 2>는 y축의 에지의 픽셀이고 그라디언트 방향은 x축을 바라보고 있다. A지점에서 그라디언트의 값이 B, C보다 큰지 체크하고, A가 가장 클 경우 무시하고, 작을 경우 0으로 처리한다.

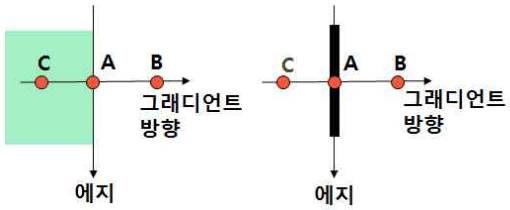

그림 2. 캐니 에지 검출 알고리즘 구조[7]

문턱 값 판단은 검출된 에지의 실 유무를 판단하는 단계로, 문턱 값의 최솟값과 최댓값을 지정한 후, 최솟값과 최댓값 사이에 있는 픽셀들은 연결 구조를 확인한 후 에지를 판단한다. <그림 3>은 문턱 값을 확인하기 위한 구조를 나타내고 있다.

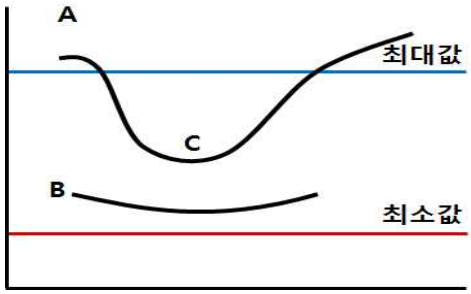

그림 3. 문턱 값을 이용한 에지 처리[7]

A는 최댓값보다 위에 있으므로 에지로 판별되며, B와 C는 최댓값과 최솟값 사이에 있는 픽셀이다. B는 A와의 연결이 끊어진 에지에 최댓값이 아니기 때문에 제거하며, C는 A와 연결이 되어있는 것이 확인되므로 에지로 판단하게 된다.

<그림 4>는 원본 사진부터 흑백처리, 가우시안 블러 처리, 캐니 에지 검출 알고리즘까지 사용한 결과를 나타내고 있다.

그림 4. 문턱 값을 이용한 에지 처리

Ⅳ. 선형 검출 알고리즘 구현

<그림 5>는 선형 검출 알고리즘을 사용한 결과를 나타내고 있다.

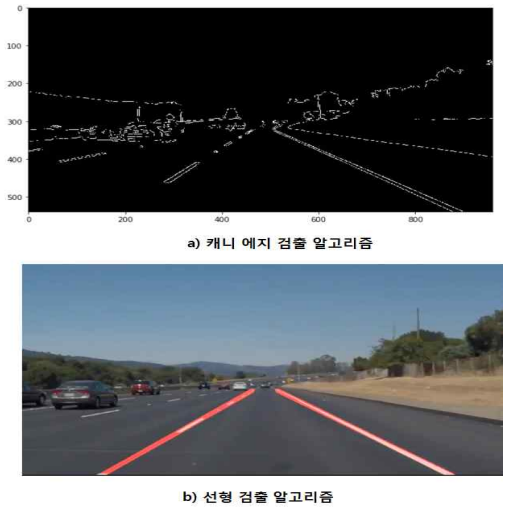

그림 5. 선형 도형 검출 알고리즘 사용

a)는 캐니 에지 검출 알고리즘을 사용하였고, <그림 5>의 b)는 a) 처리 후 선형 도형 검출 알고리즘을 사용한 후 원본 이미지와 합친 화면이다. <그림 5> b)를 보면 캐니 에지 검출 알고리즘에서 추출된 차량들이 제거되고 차선 검출이 가능함이 확인되었다.

Ⅴ. 결 론

본 논문에서는 라즈베리파이3 B+에서 OpenCV를 이용하여 선형 검출에 대한 시스템을 구현하였다. 케니 에지 검출 알고리즘을 통한 선형 검출 알고리즘이 소형컴퓨터인 라즈베리파이3 B+에서도 잘 동작하는 것이 확인되었다. 향후 과제는 자동차가 붙어있거나 간격이 좁을 경우 원하지 않는 선형 까지 추출되는 문제점을 해결하는 것을 목표로 한다.

References

[1] S.M. Lee, “The Technical Trands of Electric Vehicle Autonomous Driving”, The Korean Institute of Electrical Engineers, vol. 69(5), pp.31-35, May.2020.

[2] S.J. Jang, “Development trend of autonomous driving (smart) automobile technology”, ITFIND, vol. 1718, Oct. 2015

[3] E.S. Park, C.H. Yu, J.W. Choi, “Development of a Lateral Control System for Autonomous Vehicles Using Data Fusion of Vision and IMU Sensors with Field Tests”, Journal of Institute of Control, Robotics and Systems, vol. 21(3), pp. 179-186, Mar.2015

[4] U.H. Lee, S.Y. Yoon, I.W. Shim, S.H. Shin, J.W. Choi, J.W. Oh, H.C. Shim, I.S. Kweon, S.B. Choi, “KAIST EureCar: Development of unmanned autonomous driving vehicles capable of environmental awareness and collision avoidance in complex road conditions,” Korea Robotics Society, vol. 10(2), pp. 20-31, May. 2013

[5] opencv.org, Smoothing Images [Internet], Available: https://docs.opencv.org/trunk/d4/d13/tutorial_py_filtering.html

[6] opencv.org, Smoothing Images [Internet], Available: https://docs.opencv.org/trunk/d4/d13/tutorial_py_filtering.html

[7] opencv.org, Canny Edge Detection [Internet], Available: https://docs.opencv.org/trunk/da/d22/tutorial_py_canny.html