자동 조향 장치

(Automatic Steering System)

홍 진 문*, 구 광 모, 김 민 성, 김 유 성, 최 성 엽

(Jin-Moon Hong, Kwang-Mo Koo, Min-Seong Kim, Yu-Seong Kim, Seong-Yeop Choi)

부경대학교 전자공학과

Abstract : This paper proposes an auto steering system that prevents dangerous conditions caused by the driver's sudden manipulation by using artificial intelligence technique. The proposed system is loaded with a camera at the front of the vehicle to recognize traffic setting and the vehicle's steering position in real-time. Python OpenCV and YOLOv3 are used to identify objects on the road and to process video information. In addition, Raspberry Pi is used for handling video data and helping the automobile steer automobiles. Experimental results demonstrate that lanes and cars were successfully recognized.

Keywords : Automatic, Steer, Artificial Intelligence, Secondary Accidents, OpenCV, YOLOv3, Safety System, Raspberry Pi, Video Data Processing

Ⅰ. 서론

1.1. 연구 배경

도로에서 1차사고 발생 후, 연속하여 발생하는 2차사고의 치사율이 5배 이상 높다. 이러한 2차사고의 방지는 차량자체의 제동능력이나 운전자의 운전능력에 지대한 영향을 받고 이러한 능력은 개인별, 차량별 편차가 대단히 크다.[1]

구분		2011년	2012년	2013년	2014년
1차 사고	사고(건)	2562	2525	2419	2328
	사망(명)	218	293	220	218
	치사율	8.5	11.6	9.1	9.4
2차 사고	사고(건)	78	75	77	67
	사망(명)	47	50	44	35
	치사율	60.3	66.7	57.1	52.2

Table 1. Secondary Accidents Lethality

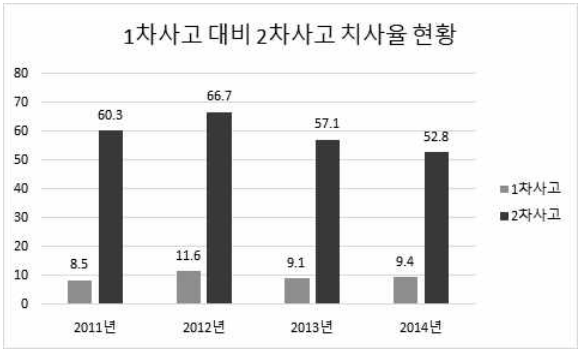

Fig 1. Status of Secondary Accident Lethality

1.2. 연구 목적

차량 전방의 도로상황을 실시간으로 AI 및 컴퓨터를 사용하여 분석한다. 분석한 결과를 바탕으로 운전자의 급조작 상황을 감지하거나, 이미 1차사고가 발생한 경우 이를 인식하여 시스템이 차량을 직접 제어한다. 운전자보다 신속한 차량제어를 통해 위험상황에서 적절하게 차량감속 및 정차 과정을 거쳐 2차 사고를 예방 및 2차사고로 인한 피해를 최소화한다.

Ⅱ. 관련 이론 및 기술

Auto Steering System은 카메라를 통하여 인식된 전방 영상을 통하여 도로 및 사물을 인식하고

이를 바탕으로 위험 상황 발생 시 차량의 조향과 가속을 제어한다(Fig 2).

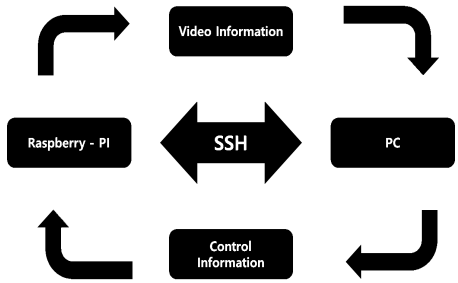

Fig 2. Process and Control

2.1. Line detection : 도로 인식

2.1.1 Camera calibration

카메라 영상으로 얻은 객체는 카메라의 위치, 사용되는 렌즈, 객체와의 거리 등에 따라 결정된다. 이 과정에서 왜곡이 일어나게 되는데 이러한 요인들을 제거하여 원래 객체의 상을 얻는 것을 Camera Calibration이라고 한다(Fig 3).

2.1.2 Perspective transform

3D 내 객체를 2D 이미지로 투영시키는 변환이다. 도로 사진을 Perspective Transform 하면 전방에서 본 시점의 이미지(Fish-Eye View)가 위에서 본 시점의 이미지(Bird-Eye View)로 변환된다(Fig 4).[3]

2.1.3 Filtering

Filtering은 여러 수식을 이용하여 이미지를 이루고 있는 픽셀 행렬을 다른 값으로 바꾸어 이미지를 변형하는 것을 의미한다(Fig 5). 그 중에서 Color Filtering은 입력받은 영상데이터에서 특정 색상의 수치 범위만 남기고 그 외의 범위는 제거한다.[4]

2.1.4 Sliding-window-search

조사창을 이용하여 원하는 객체가 있는지를 탐색한다(Fig 6).[5]

2.1.5 Super imposition

원근감이 없는 이미지를 원래 상태로 되돌리는 과정이다. Sliding-Window-Search로 인식한 차선 데이터를 사용자가 볼 수 있도록, 원본 이미지에 위의 결과를 표시한다(Fig 7).[6]

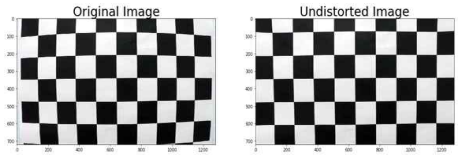

Fig 3. Camera Calibration

Fig 4. Bird-Eye View

Fig 5. Color Filtering

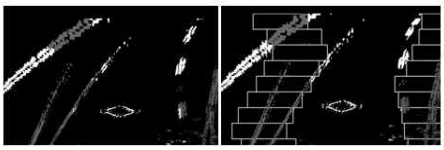

Fig 6. Sliding-Window-Search

Fig 7. Super Imposition

2.2. Object Detection : 사물 인식

2.2.1. YOLO

YOLO는 Deep CNN을 통하여 학습한 특징으로 객체를 감지하는 프로그램이다. YOLO는 하나의 신경망을 전체 이미지에 적용하는데, 사용하는 신경망 네트워크는 입력된 이미지를 여러 영역으로 나누고 각 영역에 대한 Bounding Box를 통해 확률을 예측한다(Fig 8).[7]

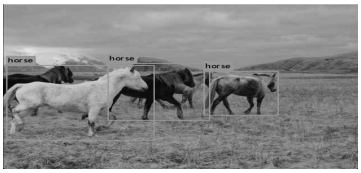

Fig 8. YOLO v3 and Bounding Box

2.2 Weight(가중치 파일)

객체를 학습시키기 위해서는 학습할 이미지와 파라미터 파일들, 가중치 파일이 필요하며 학습을 완료하면 학습시킨 객체들을 인식하는 가중치 파일이 생성된다.

Ⅲ. 제작과정1(소프트웨어)

기본적으로 시스템은 카메라를 이용하여 영상을 받아오고 Python OpenCV에서 5단계의 알고리즘을 거쳐 차선을 인식하고 YOLO V3에서 학습된 데이터를 기반으로 객체를 인식하고 그에 따라 차량을 제어하는 과정으로 이루어진다(Fig 9).

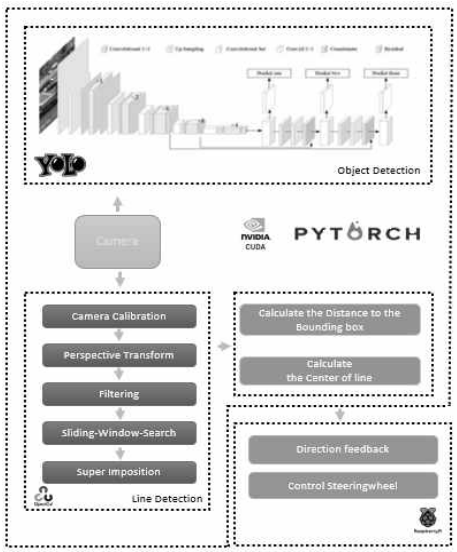

Fig 9. Processing Sequence

3.1 Line Detection : 도로 인식

카메라로 받아들인 영상의 왜곡을 제거하기 위해 cv2.getPerspectiveTransform 함수를 통해 Perspective Transform하여 Bird-Eye View로 변환시켜 차선 검출을 용이하게 한다. 변환된 프레임에서 노란색 부분과 흰색 부분만 제외 한 나머지 색을 Color filtering을 통해 제거한다. 필터링을 거친 프레임을 Sliding-Window-Search기법을 통해 노란색과 흰색부분을 검출하여 하나의 선으로 연결한다. 선 사이의 부분을 색으로 채우고 Super Imposition을 통해 원근감이 있는 원래의 이미지로 되돌린다.

3.2. Object Distinction : 객체 구분

이미지에서 각각의 객체를 검출하고 Bounding Box을 학습시킨 Weight를 통해 확률을 예측한 뒤 가장 확률이 높은 Label을 Bounding Box에 표시한다. 과도한 Bounding Box가 생성되어 오류확률이 증가하는 것을 방지하기 위해 Bounding Box 생성 한계치를 0.3으로 설정하였다.

3. Car Control : 차량 제어

충돌 방지를 위한 전방 거리를 계산하기 위해 Bird-Eye View에서 앞 차량과 카메라 사이의 픽셀의 개수를 세고 그 픽셀 숫자와 해상도에서 얻은 픽셀 당 길이를 곱하여 앞 차량과의 거리를 구한다.

Ⅳ. 제작과정2(하드웨어)

4.1. Raspberry-PI

하드웨어는 Raspberry-PI 3 기반에 서보모터 4개, 카메라 1개를 기본으로 하여 작동하게 설계되었다. 기본적으로 장착하는 카메라는 시야가 너무 협소하였기에 광각의 시야를 얻기 위해 카메라의 위치를 조정하였다.

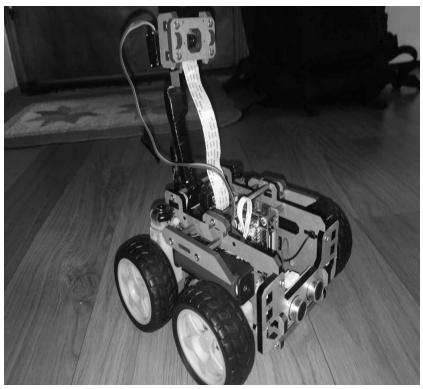

Fig 10. Raspberry-PI Hardware

Ⅴ. 작동 결과

전술한 제작과정을 거쳐 프로젝트를 최종 시연하는 모습(Fig 11)과 차량에 탑재된 객체 및 차선 인식에 대한 2회차 실행 결과이다(Fig 12). 아래 표의 내용은 카메라로부터 가장 가까운 객체의 인식 정확도를 추출한 것이다(Table 2). 그리고 객체 및 차선 인식 소프트웨어 실행의 최종 결과이다(Fig 13).

객체 인식 정확도가 99%에서 73%로 급격히 감소함을 볼 수 있는데, 이는 카메라로부터 객체까지의 거리가 일정 픽셀 상의 거리를 초과하여 발생하는 것으로 해석 할 수 있다. 픽셀 거리에 따른 정확도를 분석한 그래프(Fig 14)에 따르면 시스템은 150px에서

95%, 이를 초과하는 경우 대략 73%로 정확도가 급격히 감소함을 확인하였다. 실제로, 시스템은 거리가 약 150px인 부분에서(Fig 15) 차량을 95%의 정확도로 인식하였다. 그러나 이는 영상에서 육안으로도 구분 할 수 없을 정도의 거리로 충분히 안전에 대한 신뢰성을 보장할 수 있을 것이다. 따라서 정확도가 어느 지점에서 급격하게 떨어질 경우 물리적 거리가 상당히 멀기 때문에 중요치 않은 정보로 판단하여 시스템은 그에 맞춰 동작 할 수 있을 것이다.

Fig 11. Operation Result

시행 횟수	정확도
1회	98%
2회	99%
...	
300회	98%

Table 2. Object Recognition Accuracy (Implemented 300 Times)

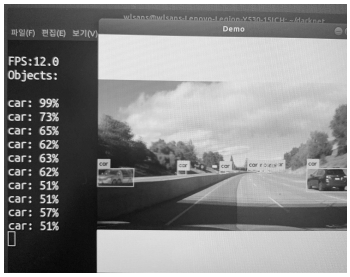

Fig 12. Object Detection Execution Results

Fig 13. Final Execution Result of Software

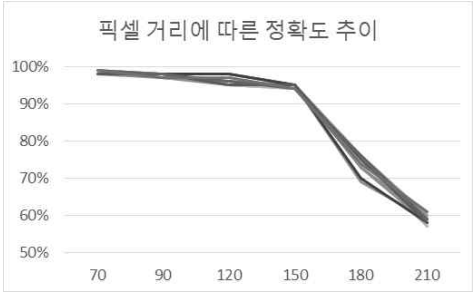

Fig 14. Accuracy Trend by Pixel Distance

Fig 15. Recognizing Accuracy Near 150 Pixel

Ⅵ. 결 론

본 논문에서는 AI를 이용한 Automatic Steering System에 대해 소개하였다. 이 시스템은 운전자의 급조작 상황이 발생했을 때 이를 인지하고 차량을 조작한다. 차량 전방의 카메라는 도로의 상황을 실시간으로 컴퓨터로 전달한다. Raspberry Pi는 차량 전방 영상을 처리 후 상황 발생 시 그에 따라 차량을 제어한다. 전술한 시스템은 차량 안전 뿐 아니라 이 시스템이 탑재되지 아니한 이미 생산된 차량에도 이식 할 수 있는 넓은 응용 범위를 가질 수 있는 시스템의 기본적인 모델이 될 것이다.

참 고 문 헌

[1] 한국도로공사

[2] https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_calib3d/py_calibration/py_calibration.html

[3] https://en.wikipedia.org/wiki/3D_projection#Perspective_projection

[4] https://blog.naver.com/hirit808/221486800161

[5] https://www.pyimagesearch.com/

[6] https://m.blog.naver.com/PostView.nhn?blogId=audiendo&logNo=220828949865&proxyReferer=https:%2F%2Fwww.google.com%2F

[7] https://pjreddie.com/darknet/yolo/